CHARACTERIZATION OF LYMPHATIC PUMP FUNCTION IN RESPONSE TO MECHANICAL LOADING

A Thesis
Presented to
The Academic Faculty

by

Jeffrey A. Kornuta

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
August 2014

CHARACTERIZATION OF LYMPHATIC PUMP FUNCTION IN RESPONSE TO MECHANICAL LOADING

Approved by:

J. Brandon Dixon, Ph.D., Advisor
Georgia W. Woodruff School of
Mechanical Engineering
Georgia Institute of Technology

Rudolph L. Gleason, Ph.D.
Georgia W. Woodruff School of
Mechanical Engineering
Georgia Institute of Technology

Jun Ueda, Ph.D.
Georgia W. Woodruff School of
Mechanical Engineering
Georgia Institute of Technology

Ajit P. Yoganathan, Ph.D.
Wallace H. Coulter Department of
Biomedical Engineering
Georgia Institute of Technology

Garrett B. Stanley, Ph.D.
Wallace H. Coulter Department of
Biomedical Engineering
Georgia Institute of Technology

Date Approved: 01 April 2014

To my friends, family, and fiancée.

ACKNOWLEDGEMENTS

I would like to thank both the National Science Foundation Graduate Research Fellowship Program and National Institutes of Health (R00HL091133 and R01HL113061) for their support in funding this research. I would also like to thank my advisor Dr. J. Brandon Dixon for his support and for giving me the opportunity to work on such an interesting project. Additionally, I would like to thank my committee for their insights and contributions.

Incredible thanks goes to my past and current lab mates: Dr. Matthew Faulkner, Dr. Matthew Nipper, Dr. Alana Reed, Mike Weiler, Jamie Huffman, Ryan Akin, and Tyler Nelson. Each of them were immensely helpful to me, and their input and/or guidance on all aspects of this project is greatly appreciated. Also, immense appreciation goes to all of my previous undergraduate students: Arina Korneva, Chris Blackburn, Emilio Salazar, Zack Danielak, Victor Yusuf, Laurissa Rybacki, Benji Hoover, Hudson Chancy, and Phillip Johnston.

I would also like to recognize my colleague Timothy Kassis for always taking the time to assist me in all aspects of this project. His generosity, selflessness, and expertise is always subtle but will never be forgotten; and I would like to take this opportunity to truly thank him for all of his contributions to my project and to LLBB. Also, I would like to thank both John Dykes and Ed Scheuermann for their help with the more technical facets of this project, and I would like to express my gratitude to Andrew Siefert for always having incredible insight and suggestions on various aspects of this research. In addition, thanks goes to Kyle French in the ME electronics shop and Mark McJunkin and Marty Jacobson in the BME machine shop for all of their help and advice.

Last but certainly not least, I would like to thank my parents, family, friends, and fiancée for their encouragement and support these past few years. I could not have arrived at this point without them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The lymphatic system is crucial for normal physiologic function, performing such basic functions as maintaining tissue fluid balance, trafficking immune cells, draining interstitial proteins, as well as transporting fat from the intestine to the blood. To perform these functions properly, downstream vessels (known as collecting lymphatics) actively pump like the heart to dynamically propel lymph from the interstitial spaces of the body to the blood vasculature. However, despite the fact that lymphatics are so important, there exists very little knowledge regarding the details of this active pumping. Specifically, it is known that external mechanical loading such as fluid shear stress and circumferential stress due to transmural pressure affect pumping response; however, anything other than simple, static relationships remain unknown. Because mechanical environment has been implicated in lymphatic diseases such as lymphedema, understanding these dynamic relationships between lymphatic pumping and mechanical loading during normal function are crucial to grasp before these pathologies can be unraveled. For this reason, this thesis describes several tools developed to study lymphatic function in response to the unique mechanical loads these vessels experience both *in vitro* and *ex vivo*. Moreover, this work investigates how shear stress sensitivity is affected by transmural pressure and how the presence of dynamic shear independently affects lymphatic contractile function.

CHAPTER I

INTRODUCTION AND LITERATURE SURVEY

1.1 Background and Motivation

The lymphatic system was first discovered as early as 1622 by Gasparo Aselli, where he observed vessels of a milky hue in the mesentery of a well-fed dog. Since then, researchers and clinicians have recognized the great importance of the lymphatics in fundamental physiology and supporting some of the body's most critical functions. These functions include maintaining tissue homeostasis (*i.e.* fluid balance), removing particulate matter from the interstitium, trafficking immune cells, and transporting lipid from the intestine to the blood [62]. On a higher level, the lymphatic system acts like a one-way transport mechanism for both fluid and proteins from the interstitial spaces of the body's tissues to return to the blood circulation. As such, the lymphatics act like an extensive sewer system [Fig. 1.1], returning excess proteins and plasma fluid from capillaries that were not taken back up by the venous portion of the vasculature. The importance of the lymphatic system in this process may be seen directly through the Starling equation, which describes the amount of fluid transported across the capillaries in the interstitial spaces:

$$J_v = K_f\left[(P_c - P_t) - \sigma_d(\pi_c - \pi_t)\right] \tag{1.1}$$

where J_v is the rate of net transcapillary fluid movement (filtration when positive and absorption when negative), K_f is the filtration coefficient (representing hydraulic conductance), $P_c - P_t$ is the transmural pressure, σ_d is the osmotic reflection coefficient, and $\pi_c - \pi_t$ is the osmotic transmural pressure. Because of the osmotic pressure component from interstitial proteins, the capillary bed has a positive net fluid flux into the interstitium; thus, the blind-ended lymphatic vessels in these spaces—termed the initial lymphatics—ultimately preserve tissue homeostasis by removing these macromolecules and excess fluid.

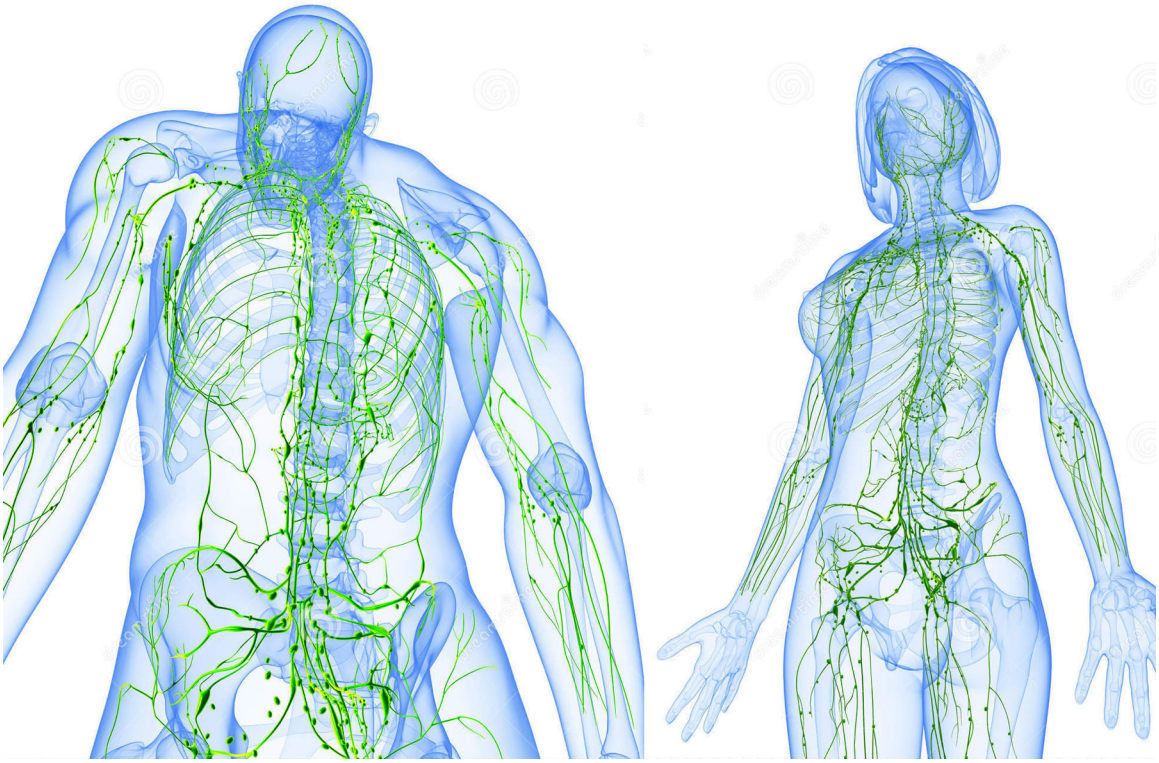


Figure 1.1: High-level anatomy of the lymphatic system. Royalty-free photos courtesy of `dreamstime.com`.

Further downstream of the initial lymphatics, the network continuously merges in to larger vessels called the collecting lymphatics. Like the initial lymphatics, the collecting vessels consist of luminal lymphatic endothelial cells (LECs); however, they are also surrounded by a thin layer of muscle cells that allow these vessels to actively contract and propel fluid toward the venous circulation. Similar to veins, the collecting lymphatics are separated into unit segments termed lymphangions by one-way valves to help ensure unidirectional transport of lymph to its final destination at the left subclevian vein [67] [Fig. 1.2]. Coupled with the initial lymphatic vessels that are anchored to the extracellular matrix (ECM) and have loose cell junctions [Fig. 1.3], these one-way valves in the collecting lymphatics also encourage lymph transport due to the effects of external factors such as respiration, skeletal muscle contractions, and interstitial fluid formation. In other words, due to the unique anatomical structure of both the initial and collecting lymphatics,

extrinsic factors arising from normal physiologic function actually promote passive lymph transport and tissue homeostasis.

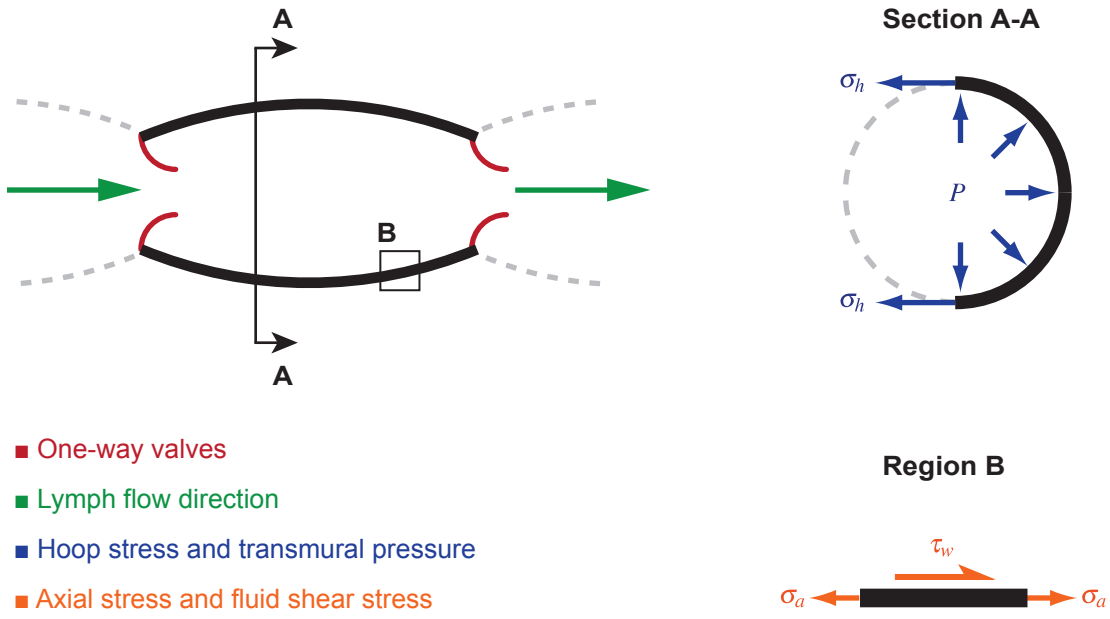


Figure 1.2: Collecting lymphatics are grouped into individual pumping units called lymphangions, where they are separated by one-way valves. These vessel segments are exposed to a variety of time-varying mechanical forces, including shear stress, τ_w, on the endothelium due to fluid flow and circumferential (hoop) stress, σ_h, due to transmural pressure, P.

Like veins, the muscle layer surrounding the collecting lymphatics allow these vessels to actively alter their tone to change their inherent resistance to fluid flow. In this way, these vessels may become better fluid conduits in response externally applied pressure gradients. However, extrinsic transport factors alone are unable to transport lymph back into the blood circulation, which recent estimates put at almost 8 L/day for humans [38]. Unlike most veins, these vessels can rapidly contract to quickly eject fluid from one lymphangion to another, sometimes up to 75% of their diameter [11, 44]. Like in the heart, these rapid phasic contractions are crucial in lymphatic transport since interstitial fluid pressure alone is insufficient to move lymph against the adverse pressure gradient present in the system [2]. Thus, the collecting lymphatics display a unique and intrinsic pump-conduit duality

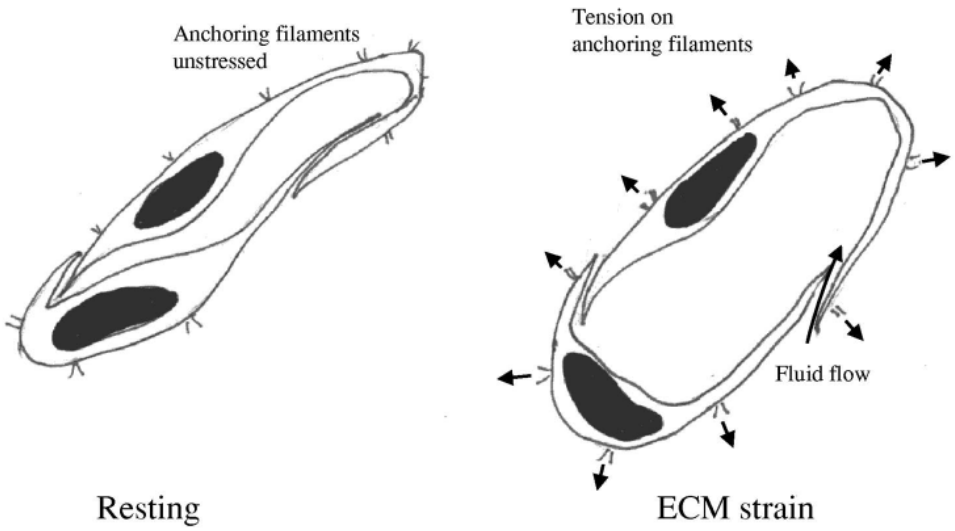


Figure 1.3: Because the LECs in the initial lymphatics are attached to the ECM with anchoring filaments and have loose junctions, expansion of the ECM promotes fluid flux into the lymphatic system. Diagram from Swartz, 2001 [62].

in their role of transporting lymph: they tonically adapt their diameter to become better conduits (like veins), yet they also phasically contract to pump lymph fluid forward (like the heart).

One of the primary reasons this intrinsic duality can exist stems from the collecting vessels' ability to sense changes in their local environment. Specifically, the contraction dynamics of the collecting lymphatics have been shown to be sensitive to how they are mechanically loaded [40, 19, 9] [Fig. 1.2]; thus, it has been hypothesized that the lymphatics' sensitivity to the local mechanical environment aids in optimizing lymph transport [19]. This ability to sense and respond to local mechanical forces would certainly be beneficial to the vessels' intrinsic pumping, which must actively adapt to the rapidly varying loads experienced during normal, physiologic circumstances [36]. For instance, like the heart, collecting lymphatic vessels have been shown to quickly react to different levels of transmural pressure [40, 24] and preload/afterload [59, 10]—parameters that change continuously based on levels of lymph formation. In addition, the rate of applied transmural pressure has also been shown to alter pump function [9]; because these rate-sensitive

changes were shown to be different to that of the portal vein, it may be possible that the lymphatics have adapted to compensate for the rapidly varying changes in load that can occur *in vivo*.

In addition to depending on transmural pressure and similar to blood vessels, studies on the contraction dynamics of isolated lymphatic vessels have also demonstrated a dependency on the fluid shear stress applied within the lumen. Specifically, an externally applied pressure gradient resulted in the inhibition of lymphatic pumping amplitude, frequency, and tone in response to luminal fluid shear stress [19, 18]. In these studies, higher magnitudes of applied transaxial pressure gradient resulted in decreased pumping amplitude, decreased pumping frequency, and decreased tone. Additionally, these responses have been shown to be mediated by the lymphatic endothelium and to use similar mechanisms that regulate vasoactive responses in the blood vasculature such as nitric oxide (NO) and endothelin-1 (ET-1) [31, 18, 20, 58]. In all, the contractile responses of the collecting lymphatics to fluid shear stress supports the notion of optimizing lymph transport through the pump-conduit duality: becoming more like a conduit when introduced to incoming flow, and behaving more like a pump in other cases. This hypothesis is further supported by Gashev *et al.*, who showed that isolated vessels displayed different sensitivities to shear stress based on anatomical location. For instance, vessels from the mesentery, which are typically exposed to enhanced levels of lymph formation due to intestinal absorption, were found to be much less sensitive to contractile inhibition when exposed to shear stress compared to a much more downstream vessel like the thoracic duct [18].

In conjunction with applied wall shear stress due to fluid flow, the transition of flow direction has also been shown to mediate lymphatic contractility. Specifically, Gashev *et al.* demonstrated in an isolated vessel that rapidly changing from orthograde to retrograde flow temporarily increased contraction frequency in isolated lymphatic vessels [19] [Fig. 1.4]. This result was also the case when the flow direction changed from orthograde or retrograde flow to no flow. In this study, they proposed that this fast chronotropic response

could be a useful physiologic mechanism to prevent unneeded inhibition of lymphatic contractions during regular pumping activity. Because lymphatic flow rates have been shown to be rapidly time-varying and oscillatory during normal, physiologic function [11], this blocking effect could further promote lymph transport in cases where contraction inhibition would actually decrease lymph transport efficiency within a chain of vessels. However, no one has been able to verify that this fast chronotropic response exists within an isolated collecting lymphatic exposed to a continuous *in vivo*-like waveform while controlling for the confounding effects of average transmural pressure. Moreover, since shear stress has previously been shown as a self-regulatory element in isolated lymphatic vessels [19, 20], this dynamic pressure gradient could play an essential role to the contractile coordination observed among lymphangions [68]—a possibility which could greatly enhance lymph transport efficiency. However, no one has demonstrated that the transaxial pressure gradient (hence fluid shear stress) is able to dynamically coordinate contractile activity independent of transmural pressure.

With respect to the magnitude of fluid shear stress in lymphatics, the levels of shear experienced by LECs *in vivo* are much less compared to similar-sized blood microvessels [24, 11]. This difference in sensitivity could be attributed in part to a marker such as vascular endothelial growth factor receptor-3 (VEGFR-3), a growth factor receptor expressed differentially on LECs; however, very little is known about the factors mediating the shear stress sensitivity of LECs. In addition to a protein such as VEGFR-3, one potential candidate could be the transmural pressure exerted on the vessel. Given that isolated blood vessel studies have previously shown interplay between intraluminal pressure-induced responses with flow-induced dilations [34, 60], this coupling could be an important factor in normal contractile function given the enhanced mechanical sensitivity of lymphatics [39]. Additionally, blood endothelial cell responses to fluid shear stress are known to be partly mediated by junctional proteins (specifically, VE-cadherin, PECAM-1, and VEGFR-2) [63], with fluid shear stress recently being shown to increase the force on the junctional protein

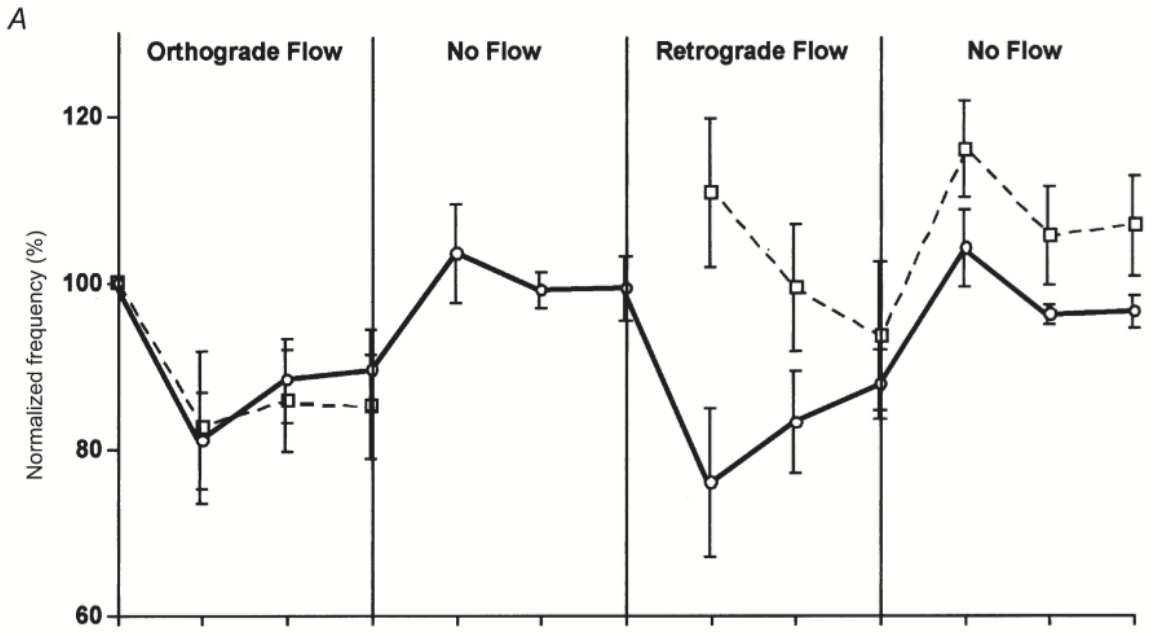


Figure 1.4: A rapid change in flow direction temporarily obstructs the typical flow inhibition of contraction frequency seen within the collecting lymphatics. Figure from Gashev *et al.*, 2002 [19].

PECAM-1 [7]. All together, this evidence suggests that the transmural pressure within a collecting lymphatic, which certainly affects circumferential stress within the endothelium, and thus the force exerted on the junctional proteins, could assist in mediating the vessel's shear sensitivity.

These relationships that have been observed between lymphatic contractile function and mechanical loading is particularly interesting in the context of lymphatic pathologies that alter the mechanical environment surrounding the vessels. One such pathology is lymphedema, a disease characterized by gross swelling of the tissue estimated to affect over 130 million people worldwide [54, 55]. Clinically, lymphedema often manifests itself in the extremities [Fig. 1.5], and there currently exists no effective cures or therapies for patients. Mechanically, lymphedema is likely to result in elevated transmural pressure in the effected area [23], and the accumulation of fibrotic tissue and adipocytes due to lymph stasis can

drastically alter the mechanical environment surrounding the vessels [56]. In addition, patients that survive congenital heart defects such as those who undergo a Fontan procedure often display elevated venous pressures and lymphatic dysfunction [21]. All together, this evidence certainly suggests that mechanics plays an integral role in both disease onset progression; however, it is crucial to first understand the role of mechanics during normal lymphatic function before researchers can unravel its potentially important implications in disease. Moreover, because the state-of-the-art in treating lymphedema—compression garments and manual massage—is inherently a mechanical process, better understanding the basic biomechanics of the lymphatics could greatly enhance and further facilitate advancements in lymphedema treatment.

Figure 1.5: Severe advanced lymphedema with fibrosis, hyper pigmentation, and excoriation of the skin (from Rockson, 2001 [54]).

One of the reasons these basic questions in lymphatic biomechanics remain unanswered is the lack of adequate experimental tools. Tools and techniques do exist for studying and quantifying biomechanical characteristics of the lymphatics *in vivo*, such as through the observation of fluid flow and contraction dynamics in the exposed rat mesentery (invasive) [11, 30], or through the quantification of transport and functional metrics after the injection of a near-infrared (NIR) dye (non-invasive) [37, 66, 65, 45]. However, *in vivo* techniques

such as these simply permit observation of the mechanical phenomena present and do not allow precise control of the imposed mechanical conditions. For instance, Dixon *et al.* was able to quantify lymphatic transport characteristics such as flow rate and contraction dynamics during normal function *in vivo* through observing (optically via brightfield microscopy) the exposed mesentery of a rat [11]. Although this study was able to provide insight for the first time on the magnitudes and dynamic characteristics of lymph flow (with corresponding wall shear stresses) in relation to the contractile properties of the vessel, the inability to impose specific mechanical conditions onto the vessels studied certainly limited the amount of information that could be gathered. Hence, the ability to design deliberate experiments with variable and controlled conditions relating imposed mechanical loads on lymphatic contractile dynamics remains extremely limited in current *in vivo* experimental setups. Consequently, to gain further insight into how the biomechanical environment governs lymphatic contraction dynamics, researchers have used other experimental approaches—namely, *in vitro* and *ex vivo* techniques—in order to more properly study these phenomena.

In order to study the specific effects of fluid shear stress on LECs *in vitro*, investigators have developed an assortment of devices for exposing endothelial cells (ECs) to dynamic shear stress waveforms [25, 48, 15]. However, these systems are custom-built and fairly complex, rendering them largely inaccessible to the broader EC community. Commercial systems are available for applying fluid shear stress waveforms (*e.g.* Flexcell®, Ibidi, Vacu-Cell™); however, these commercial systems are quite expensive and have limited flexibility for applying various dynamic flow rates. In particular, these systems are incapable of producing anything other than small subset of pulsatile or oscillatory waveforms (*e.g.* a square wave or sine wave). Thus, in order to more effectively study lymphatic biomechanics *in vitro*, a more effective platform is needed in order to study the complex, rapidly-varying mechanical loads seen *in vivo*.

In addition, current *ex vivo* experimental devices used in isolated lymphatic or blood

vessel studies are also inadequate to study the complex behavior of the intrinsic pump to dynamic mechanical loads. The most widely-used setup generally consists of two hydrostatic pressure columns separated by an isolated vessel [40, 34, 19, 49], which allows the experimenter to statically (or in a step-wise fashion) impose a pressure gradient and transmural pressure via the height of the columns. Others have developed more complex devices capable of imposing either a sinusoidal [47] or ramped intraluminal pressure [9] in isolated lymphatic vessels; however, none of these systems has the ability to dynamically adjust both the pressure gradient (which affects shear stress via flow rate) and average transmural pressure (which affects circumferential stress) simultaneously and arbitrarily. Likewise, investigators have constructed a wide array of *ex vivo* perfusion systems to impose varying flow rate and transmural pressure waveforms on isolated blood vessels [26, 6, 22, 1, 14, 50], but none can impose arbitrary pressure gradient and transmural pressure waveforms both concurrently and independently. A system with these control capabilities would be critically important in studying the physiological effects of biomechanical stimuli on lymphatic contractility and would allow single-factor studies to be performed that would be nearly impossible with *in vivo* models.

1.2 Research Goals

The overall research goals of this thesis can be separated into two primary objectives:

1. Developing tools to study lymphatic function in response to the unique mechanical loads these vessels experience both *in vitro* and *ex vivo*, and

2. Elucidating the dynamic response of the intrinsic lymphatic pump to mechanical stimuli, specifically dynamic fluid shear stress and transmural pressure.

These objectives are broken into three primary aims (Chapters 2, 3, and 4), with specific goals listed as follows:

- Develop a multi-purpose and low-cost electronics platform for studying lymphatic biomechanics *in vitro*.

- Design, model, and build an *ex vivo* lymphatic perfusion system capable of independently controlling the two primary mechanical stimuli imposed on a lymphatic vessel: average transmural pressure, P_{avg}, which affects circumferential (hoop) stress; and transaxial pressure gradient, ΔP, which affects fluid shear stress via flow rate.

- Conceive and implement a control scheme to achieve independent control of transaxial pressure gradient and transmural pressure in embedded hardware.

- Quantify shear stress sensitivity in the collecting lymphatics (specifically, the rat thoracic duct) and determine if and how it is affected by transmural pressure.

- Substantiate the idea that continuously-varying pressure gradient (*i.e.* fluid shear stress) waveforms within a physiologic frequency range can obstruct the typical flow-induced inhibitory effect on contraction frequency while controlling for average transmural pressure.

- Determine whether or not a dynamic fluid shear stress has the capability to independently coordinate lymphatic contractile activity (*i.e.* while controlling for average transmural pressure).

1.3 Thesis Outline

- Chapter 2 will describe a low-cost, reliable microcontroller platform that was developed to study lymphatic biomechanics *in vitro*. The platform's capabilities are demonstrated through two applications: the augmentation of a commercially available pump to allow a wide variety of flow rate waveforms to be produced, and the support of a custom-built cell straining device capable of producing oscillatory strains with varying amplitudes and frequencies.

- Chapter 3 will detail an *ex vivo* lymphatic vessel perfusion system that was designed to independently control the imposed transaxial pressure gradient and average transmural pressure on an isolated lymphatic vessel using a linear, explicit model predictive control (MPC) algorithm implemented in custom hardware. In addition, a *post hoc* method of estimating both the flow rate through the vessel and fluid wall shear stress over multiple, long time windows is also described.

- Chapter 4 will report the affects of dynamic shear stress on collecting lymphatics while controlling for transmural pressure, including (1) how the shear sensitivity is affected by transmural pressure; (2) if a dynamic shear stress suppresses the typical shear-induced inhibitory effect on contraction frequency; and (3) whether or not a dynamic shear stress can independently coordinate lymphatic contractile activity.

- Chapter 5 will summarize the primary conclusions and contributions of this thesis, as well as describe future research directions that could be taken.

CHAPTER II

LOW-COST MICROCONTROLLER PLATFORM FOR STUDYING LYMPHATIC BIOMECHANICS IN VITRO

2.1 Abstract

The pumping innate to collecting lymphatic vessels routinely exposes the endothelium to oscillatory wall shear stress and other dynamic forces. However, studying the mechanical sensitivity of the lymphatic endothelium remains a difficult task due to limitations of commercial or custom systems to apply a variety of time-varying stresses *in vitro*. Current biomechanical *in vitro* testing devices are very expensive, limited in capability, or highly complex; rendering them largely inaccessible to the endothelial cell biology community. To address these shortcomings, the author proposes a reliable, low-cost platform for augmenting the capabilities of commercially available pumps to produce a wide variety of flow rate waveforms. In particular, the Arduino Uno, a microcontroller development board, is used to provide open-loop control of a digital peristaltic pump using precisely-timed serial commands. In addition, the flexibility of this platform is further demonstrated through its support of a custom-built cell-straining device capable of producing oscillatory strains with varying amplitudes and frequencies. Hence, this microcontroller development board is shown to be an inexpensive, precise, and easy-to-use tool for supplementing *in vitro* assays to quantify the effects of biomechanical forces on lymphatic endothelial cells.

2.2 Introduction

Collecting lymphatic vessels contract much like the heart to promote flow, with transport occurring through the coordinated intrinsic pumping of individual vessel segments known

as lymphangions [67, 46]. These contractions continuously expose luminal lymphatic endothelial cells (LECs) to oscillatory wall shear stress (with frequencies typically below 1 Hz) [11] as well as other forces, all of which have been shown to affect vessel pumping [24, 19, 9]. In addition, several extrinsic factors also work to promote lymph flow such as skeletal muscle contraction, respiration, and interstitial fluid formation. Because of the numerous mechanisms that affect lymph transport *in vivo*, flow rates can vary quite drastically under physiologic conditions; accordingly, the intrinsic lymphatic pump must actively respond to these varying *in vivo* loads. Many of these responses are, in part, endothelium mediated, and utilize similar mechanisms that have been known to regulate vasoactive responses in the blood vasculature [19, 31, 4]. However, the magnitudes of these forces experienced by LECs *in vivo* are significantly less than those in similar-sized blood microvessels [Table 2.1] [24, 39, 11], and recent evidence suggests that their unique biomechanical environment is essential in guiding lymphatic development [5, 57]. Moreover, the ability to apply custom time-varying shear stresses is useful in the context of elucidating the functional response of LECs, which experience rapid and often irregular changes in the mechanical environment *in vivo*.

To study the specific effects of fluid shear stress on LECs *in vitro*, investigators have developed an assortment of devices for exposing endothelial cells (ECs) to dynamic shear stress waveforms [25, 48, 15]. However, these systems are custom-built and fairly complex, rendering them largely inaccessible to the broader EC community. Commercial systems are available for applying fluid shear stress waveforms (*e.g.* Flexcell®, Ibidi, VacuCell™); however, these commercial systems are expensive and have limited flexibility for applying various dynamic flow rates. In particular, these systems are incapable of producing anything other than small subset of pulsatile or oscillatory waveforms (*e.g.* a square wave or sine wave). Thus, the author reports a technique for enhancing the capabilities of commercially-available pumps to generate a wide variety of flow waveforms, including

those seen in lymphatics *in vivo*. The proposed system is a reliable, inexpensive, and easy-to-use platform for augmenting a digital peristaltic pump with an open-loop control system to produce time-varying flow rates. Additionally, the author further demonstrates the flexibility of this platform by using it to support a custom-built cell-straining device capable of producing oscillatory strains with a multitude of amplitudes and frequencies.

Table 2.1: Typical pressure and wall shear stress (WSS) values of the body's microvessels [24, 39, 11].

Parameter	Arterioles	Venules	Capillaries	Collecting Lymphatics
Pressure (mmHg)	75–100	20–30	30–40	3–4
Average WSS (dynes/cm^2)	50–60	10–30	20–50	< 1

2.3 Methods and Results

2.3.1 Arduino Uno Development Board

The Arduino Uno development board is based on the Atmel ATmega328, an 8-bit, 16 MHz microcontroller with 14 digital input/output (I/O) pins, 6 of which are capable of pulse-width modulation (PWM), as well as a 6-channel, 10-bit analog-to-digital converter. Digital communication capabilities include UART TTL serial, SPI serial, and two-wire interface serial (I^2C). The Arduino development platform features a cross-platform, Java-based IDE as well as a C/C++ library which offers high-level access to hardware functions.

2.3.2 Programmable Fluid Shear Stress Via Peristaltic Pump

Figure 2.1 shows the flow system used to generate time-varying fluid shear stresses on an LEC monolayer grown in a parallel-plate flow chamber using an Ismatec REGLO Digital MS-4/12 peristaltic pump (IDEX Health and Science, Glattbrugg, Switzerland). After entering a desired periodic shear stress function and experimental duration via a custom Python script [Appendix A.1], the required peristaltic pump speed commands are sent serially to the Arduino Uno. The Arduino Uno stores these commands on its local EEPROM,

which allows experiments to be conducted entirely with the Arduino Uno development board. Once running, the Arduino Uno recreates the desired waveform by sending each of the locally-stored pump commands at specific time increments to the peristaltic pump via a TTL to RS232 level shifter (SparkFun Electronics, Boulder, CO). The experiment progress is displayed on an LCD, which is operated with Arduino's LiquidCrystal library.

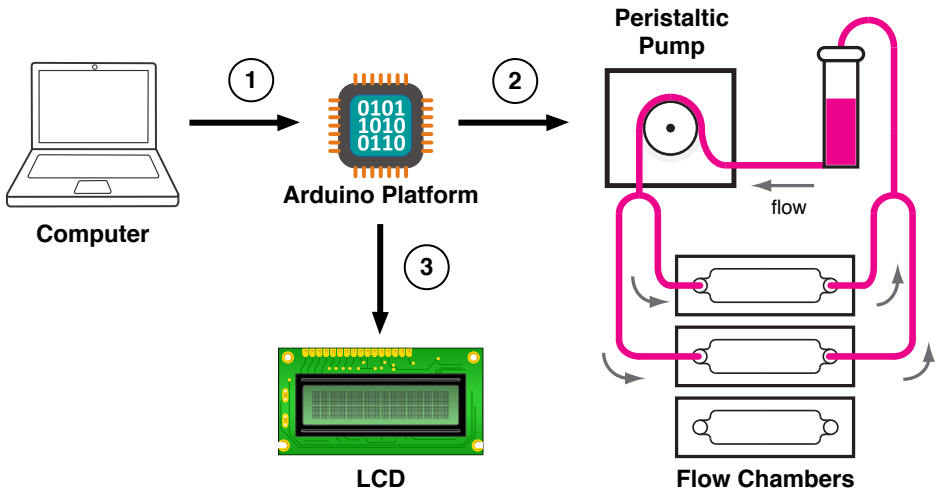


Figure 2.1: Outline of the flow system: (1) the Arduino platform is programmed by a personal computer (PC) over USB to create a desired fluid shear stress (flow rate) waveform for a certain duration of time; (2) once programmed, the Arduino can independently control the peristaltic pump via an RS232 serial connection with precise timing in an open-loop fashion; (3) the experimental progress is displayed on a connected LCD screen for the experimenter to monitor.

In order to mimic the user-desired waveform, the Python script generates a certain number of discrete pump commands based on a suitable time delay for the pump. In this way, the Arduino Uno will avoid sending commands too fast for the peristaltic pump to implement. Upon initialization of the experiment, the Arduino Uno will automatically refine this delay time using a self-calibration routine that takes extraneous pump commands (*e.g.* `start` and `change direction`) into account [Appendix A.1]. In this way, the overall period of the desired waveform is ensured to be accurate. Figure 2.2a-d shows several examples of desired flow rate waveforms compared with the measured flow rate waveforms on the peristaltic pump. The measured flow rate is obtained by acquiring the speed of the

pump rollers via an internal pump encoder which returns a square wave whose frequency is proportional to the speed. Accordingly, because a peristaltic pump is a positive displacement pump, the velocity of the rollers is linearly proportional the mean flow rate through the corresponding tube (according to its internal diameter). Overall, good agreement is achieved between the desired and measured flow rates.

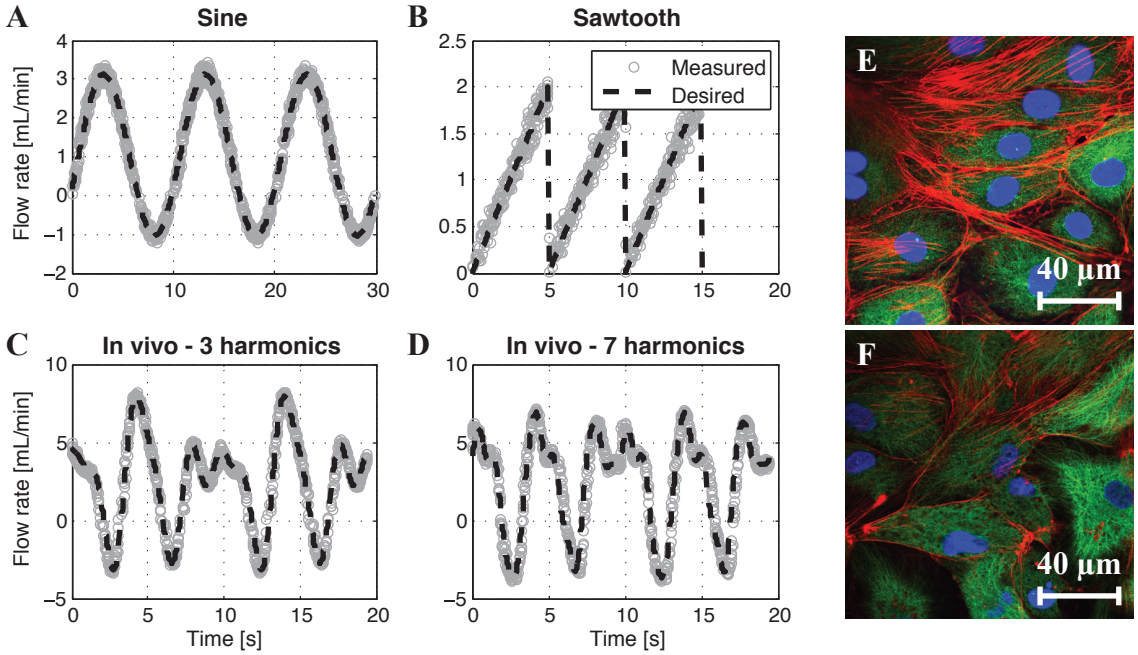


Figure 2.2: Measured vs. desired flow rates for the digital peristaltic pump being controlled by the Arduino Uno (a-d). The pump system accurately produces several desired waveforms: (a) a sine wave, (b) a sawtooth wave, (c) a waveform with the top three harmonics of a rat lymphatic waveform measured *in vivo*, and (d) a waveform with the top seven harmonics of a rat lymphatic waveform measured *in vivo* [30]. (e) Controlled LECs cultured under a static, no-flow condition. (f) LECs exposed to the sine wave found in (a) for 24 hours. For the false-colored images: the red channel is F-actin (phalloidin), the green channel is tubulin (α-β tubulin), and the blue channel is the cell nucleus (DAPI).

2.3.3 Adjustable Cell Monolayer Straining

The author has also developed a cell-straining device [Fig. 2.3a] capable of imposing a static or oscillatory strain of varying magnitude and frequency [Fig. 2.4]. The amplitude of strain may be adjusted by adjoining one end of the connecting rod to the hub at points of varying radii, while the frequency of strain may be adjusted by regulating the speed of

the DC motor powering the hub. The details of this displacement may be seen through Eq. (2.1),

$$x(t) = R - R\cos(\omega t) + L - \sqrt{L^2 - [R\sin(\omega t)]^2} \tag{2.1}$$

where $x(t)$ is the displacement of the membrane, R is the radius of the crank, ω is the rotational velocity of the crank, and L is the length of the connecting rod. Velocity feedback of the hub (crank) is acquired with an infrared photo interrupter (SparkFun Electronics, Boulder, CO), which outputs a rising voltage edge upon sensing one of the twelve slots located around the outer perimeter of the hub. Time between rising edges is measured by the Arduino Uno, which can then calculate the speed for display in revolutions per minute (rpm) on the attached LCD screen [Appendix A.2].

2.4 Discussion

In order to create time-varying waveforms on a digital peristaltic pump, such as the one used in this study, one could use a personal computer (PC) to send out serial (flow rate) commands at sufficiently-fast intervals to approximate the desired waveform. However, typical PC operating systems are not adept at precise timing, and PCs can be difficult to operate near or inside of cell culture incubators. Conversely, another option would be a conventional microcontroller: the inherent simplicity of these embedded systems give them very desirable timing characteristics and thus have been useful in a wide variety of biomedical applications [61, 3, 16]. However, this raw simplicity in hardware comes at the expense of writing the required software, which is typically written in either assembly or C/C++ and often requires intimate knowledge of the particular hardware architecture. This complexity in writing the software makes it difficult for non-experts to pick up and use these systems without a considerable time investment. Due to these limitations, the author has chosen to use the Arduino Uno development board ($30, `http://arduino.cc`), which is an open-source electronics prototyping platform based on user-friendly hardware and

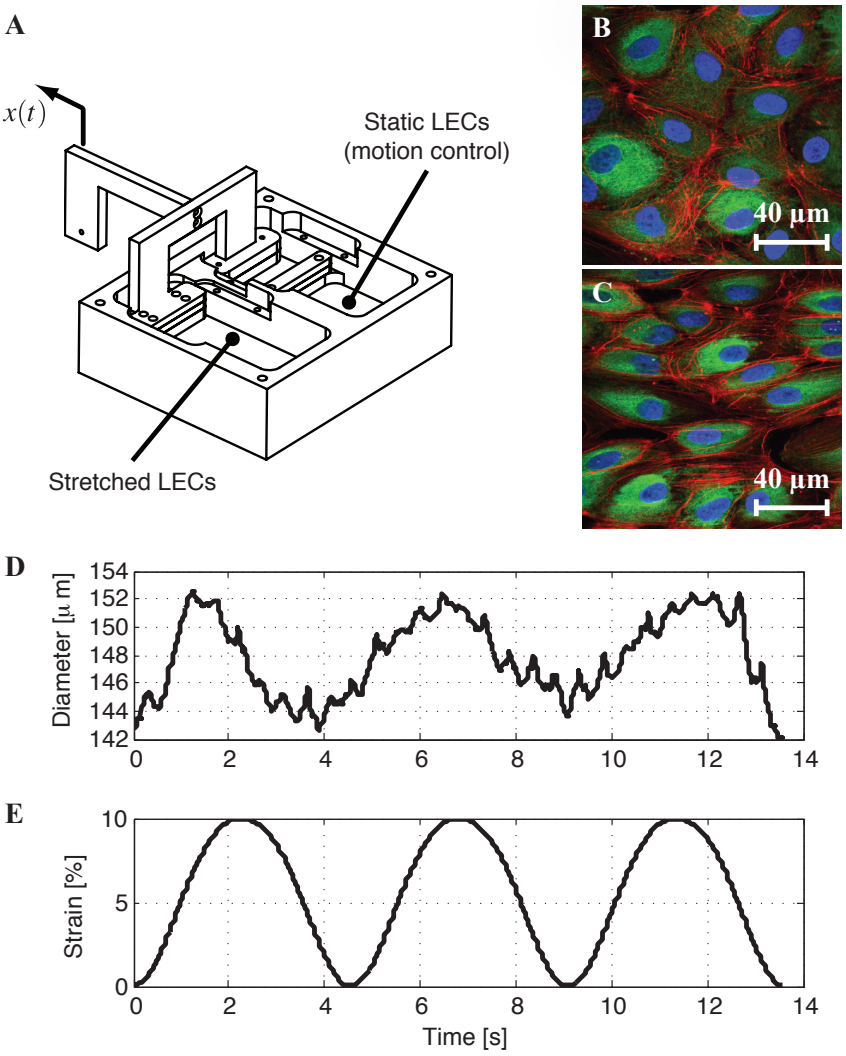


Figure 2.3: (a) Dual compartment stretch chamber based on Ives *et al.*[29]: one strains a membrane with LECs and the other moves fluid media over an LEC-covered static membrane as a motion control. (b) LECs cultured under the static (motion control) condition. (c) LECs cultured under the strain condition for 24 hours. For the false-colored images: the red channel is F-actin (phalloidin), the green channel is tubulin (α-β tubulin), and the blue channel is the cell nucleus (DAPI). (d) Diameter over time of a rat collecting lymphatic vessel measured *in vivo* [30]. (e) Example strain waveform that can be produced by the device based on Eq. (2.1) with a similar frequency found in (d).

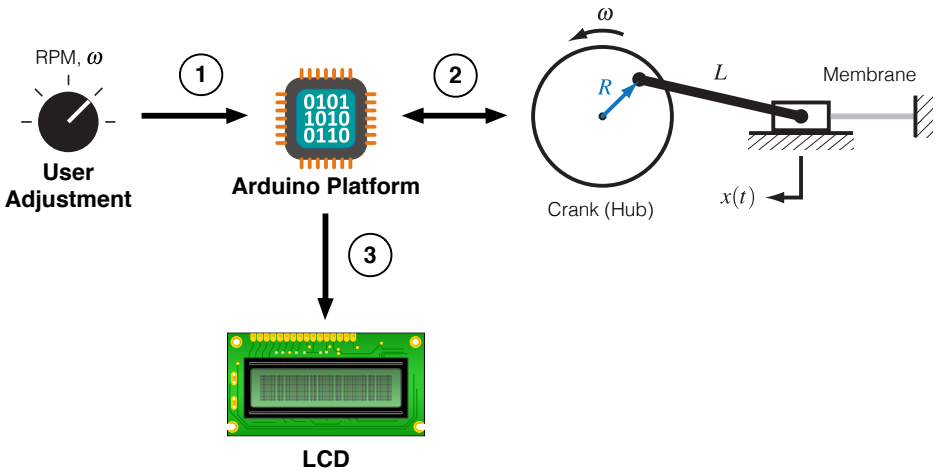


Figure 2.4: Outline of the cell-straining device: (1) the user adjusts a potentiometer knob that controls the PWM frequency output of a motor driver board; (2) the PWM frequency from the motor driver board controls the speed of the crank, which dictates the frequency of strain, and this measurement is fed back to the Arduino Uno via a photo interrupter; (3) the Arduino Uno uses the photo interrupter as an encoder to calculate the speed of the crank then displays it on an attached LCD.

software. Although based on an 8-bit Atmel microcontroller, the Arduino Uno offers high-level libraries that allows programming to be more intuitive and does not require familiarity with the low-level hardware registers usually required for I/O. In fact, its ease-of-use has made the Arduino platform recently popular with artists, hobbyists, and students interested in electronic automation [35].

The peristaltic pump system demonstrated in this study is an example of a successful augmentation with the Arduino Uno. Although the minimum delay time between commands (∼30–50 ms) required by this pump prohibits waveforms with high temporal resolutions to be used, this system can adequately reproduce the oscillatory flow functions seen in the lymphatics [Fig. 2.2c-d]. However, this limitation is unique to digital pumps such as these; pumps with an analog input would be limited only by the speed of their electromechanical dynamics, which could allow various vascular waveforms to be produced. Analog-capable pumps could make use of the Arduino Uno's PWM output and an inexpensive H-bridge circuit to produce time-varying flows.

In addition to augmenting pump-based systems, the author further demonstrates the flexibility of this platform by supporting a custom-built cell-straining device for mimicking the strain amplitudes and frequencies experienced by LECs within collecting lymphatics. The device can produce strains between 10–50% and frequencies exceeding 1 Hz, which is within the observed range for the lymphatics [24, 11, 9]. The Arduino Uno's precise timing capabilities [8], coupled with its ability to use hardware interrupts, allows the system to measure the strain frequency and display it on an attached LCD for user feedback. Thus, the author proposes the Arduino Uno as an inexpensive, flexible, and adaptable platform for the development of *in vitro* assays to quantify the effects of biomechanical forces on the lymphatic endothelium.

CHAPTER III

EX-VIVO LYMPHATIC PERFUSION SYSTEM FOR INDEPENDENTLY CONTROLLING PRESSURE GRADIENT AND TRANSMURAL PRESSURE IN ISOLATED VESSELS

3.1 Abstract

In addition to external forces, collecting lymphatic vessels intrinsically contract to transport lymph from the extremities to the venous circulation. As a result, the lymphatic endothelium is routinely exposed to a wide range of dynamic mechanical forces, primarily fluid shear stress and circumferential stress, which have both been shown to affect lymphatic pumping activity. Although various ex-vivo perfusion systems exist to study this innate pumping activity in response to mechanical stimuli, none are capable of independently controlling the two primary mechanical forces affecting lymphatic contractility: transaxial pressure gradient, ΔP, which governs fluid shear stress; and average transmural pressure, P_{avg}, which governs circumferential stress. Hence, the author describes a novel ex-vivo lymphatic perfusion system (ELPS) capable of independently controlling these two outputs using a linear, explicit model predictive control (MPC) algorithm. The ELPS is capable of reproducing arbitrary waveforms within the frequency range observed in the lymphatics in vivo, including a time-varying ΔP with a constant P_{avg}, time-varying ΔP and P_{avg}, and a constant ΔP with a time-varying P_{avg}. In addition, due to its implementation of syringes to actuate the working fluid, a post-hoc method of estimating both the flow rate through the vessel and fluid wall shear stress over multiple, long (5 sec) time windows is also described.

3.2 Introduction

The contraction dynamics of collecting lymphatic vessels (referred to as the intrinsic pump) are not predetermined and have been shown to be sensitive to changes in how they are mechanically loaded [40, 19, 9]. Since the formation of lymph can vary widely even during normal, physiologic circumstances [36], it seems reasonable to assume that lymphatics alter their behavior to optimize lymph transport due to their sensitivity to the local mechanical environment [19]. Many of these responses are, in part, endothelium mediated, and utilize similar mechanisms that have been known to regulate vasoactive responses in the blood vasculature [31, 4]. However, the magnitudes of these forces experienced by LECs *in vivo* are significantly less than those in similar-sized blood microvessels [Table 2.1] [24, 39, 11], and recent evidence suggests that their unique biomechanical environment is essential in guiding lymphatic development [5, 57]. Additionally, vascular endothelial growth factor receptor-3, a growth factor receptor expressed differentially on LECs, has recently been implicated in the lower shear stress sensitivity of lymphatics (Schwartz, M., Vascular Biology 2013, Oct 20-24, 2013, Hyannis, MA).

One such example of this active behavioral modification may be found in the inhibition of lymphatic pumping amplitude, frequency, and tone in response to luminal fluid shear stress [19, 18]. Along with fluid shear stress, altering intraluminal (transmural) pressure in isolated lymphatic vessels has also been shown to result in differing contractile (pumping) responses [40, 24]. However, the magnitude of circumferential stress (or strain) imposed on a lymphatic vessel is not alone in affecting contraction: the rate of load applied to a vessel has also been shown to alter pump function [9]. Because these rate-sensitive changes proved to be different to that of the portal vein, it suggests that the lymphatics may have adapted to compensate for the rapidly varying changes in load that can occur *in vivo*. Moreover, the notion of a function-based pumping response is further supported by observations of lymphatic vessels exhibiting differential sensitivities to mechanical loading based on their anatomical location [18].

This clear relationship between lymphatic contractile function and mechanical loading is particularly interesting in the context of lymphatic pathologies that alter the mechanical environment surrounding the vessels. One such pathology is lymphedema, a disease characterized by gross swelling of the tissue estimated to affect over 130 million people worldwide [54, 55]. Clinically, lymphedema often manifests itself in the extremities, and there currently exists no effective cures or therapies for patients. Mechanically, lymphedema is likely to result in elevated transmural pressure in the effected area [23], and the accumulation of fibrotic tissue and adipocytes due to lymph stasis can drastically alter the mechanical environment surrounding the vessels [56]. However, although mechanics may play an integral role in lymphatic pathologies, it is crucial to first understand the role of mechanics during normal lymphatic function before we can unravel its implications in disease.

Unfortunately, current *ex vivo* experimental devices used in isolated lymphatic or blood vessel studies are inadequate to study the complex behavior of the intrinsic pump to dynamic mechanical loads. The most widely-used setup generally consists of two hydrostatic pressure columns separated by an isolated vessel [40, 34, 19, 49], which allows the experimenter to statically (or in a step-wise fashion) impose a pressure gradient and transmural pressure via the height of the columns. Others have developed more complex devices capable of imposing either a sinusoidal [47] or ramped intraluminal pressure [9] in isolated lymphatic vessels; however, none of these systems has the ability to dynamically adjust both the pressure gradient (which affects shear stress via flow rate) and average transmural pressure (which affects circumferential stress) simultaneously and arbitrarily. Likewise, investigators have constructed a wide array of *ex vivo* perfusion systems to impose varying flow rate and transmural pressure waveforms on isolated blood vessels [26, 6, 22, 1, 14, 50], but none can impose arbitrary pressure gradient and transmural pressure waveforms both concurrently and independently.

This shortcoming in capability may be attributed the inherent complexity of multi-input,

multi-output (MIMO) tracking control. Because these types of systems often have outputs with coupled dynamics, controlling these outputs to track independent dynamic signals is an extremely challenging task if using relatively simple feedback control schemes like on-off or PID control [22, 1]. Hence, more advanced controllers with predictive capabilities incorporating the system's dynamics are necessary to advance the time-varying tracking capabilities of these perfusion systems. In this regard, significant strides have been made: El-Kurdi *et. al.* reported the first system identification of an *ex vivo* vascular perfusion system to obtain a (MIMO) mathematical model that could be used for control [14]. However, to the author's knowledge no one has yet successfully included one of these models into a control scheme with MIMO tracking capabilities. This attribute of independent tracking could be especially significant in studying the lymphatics, where both intrinsic pumping and extrinsic factors (such as skeletal muscle contraction, respiration, and interstitial fluid formation) can result in widely-varying fluid loads even under normal, physiological conditions. Hence, the ability to impose arbitrary and dynamic waveforms for both transaxial pressure gradient and average transmural pressure within physiologically relevant ranges is necessary to properly study the contractile effects of lymphatic vessels subject to these assorted loads.

In order to study both the independent and coupled effects of these factors on lymphatic pump function in isolated vessels, the author proposes a novel *ex vivo* lymphatic perfusion system (ELPS) capable of controlling these two parameters within an excised rat lymphatic vessel [Fig. 3.1]. Specifically, this perfusion system is meant to independently control the two primary mechanical stimuli imposed on a lymphatic vessel: average transmural pressure, P_{avg}, which affects circumferential (hoop) stress; and transaxial pressure gradient, ΔP, which affects fluid shear stress via flow rate. The author achieved this control capability using a linear, explicit model predictive control (MPC) scheme, which is implemented simply with an embedded microcontroller. Although the use of MPC is not new [53, 17, 43], it has never been implemented in a biomedical application such as this one. In short,

a perfusion system with these control capabilities would be paramount in studying the physiological effects of biomechanical stimuli on lymphatic contractility and would allow single-factor studies to be performed that would be nearly impossible with *in vivo* models. The device and compensator designs for the proposed system are summarized henceforth.

3.3 Methodology

3.3.1 ELPS Design and Hardware

The primary control objective of the system is to track a pair of desired inputs, $\Delta P = P_1 - P_2$ and $P_{\text{avg}} = (P_1 + P_2)/2$, to be imposed on an isolated lymphatic vessel. In order to achieve this, the device [Fig. 3.1a] uses two independently-actuated glass syringes in a closed-loop configuration connected with two three-way pinch solenoid valves (Cole-Parmer, Vernon Hills, IL), connected in such a way to effectively form one four-way valve [Fig. 3.1c,d]. Thus, when one syringe has expelled a majority of its fluid, the four-way solenoid valve switches while the syringes begin to propagate in the opposite direction. In this way, the system can maintain the directionality of flow with respect to the vessel for an indefinite period of time [26]. The two glass syringes are independently actuated, which permits the precise generation of arbitrary pressure gradient and transmural pressure waveforms. Each syringe is 100 μL in volume (Hamilton Company USA, Reno, NV) and is actuated by a MX80L brushless linear stage (Parker Hannifin Corp., Rohnert Park, CA) through custom mounting hardware [Fig. 3.1b, Appendix B.1]. A ViX 250AH servo drive (Parker Hannifin Corp., Rohnert Park, CA) powers each linear stage, and each drive operates in velocity mode to internally control the velocity of the syringe plunger. A separate controller (compensator) provides a set of analog input voltages, **u**, to the servo drives, which resides in the range of ± 10 V.

The system is connected by 1/16" ID, 1/8" OD Tygon® tubing, with the exception of the small segments placed in the solenoid pinch valves that are made from silicone. In

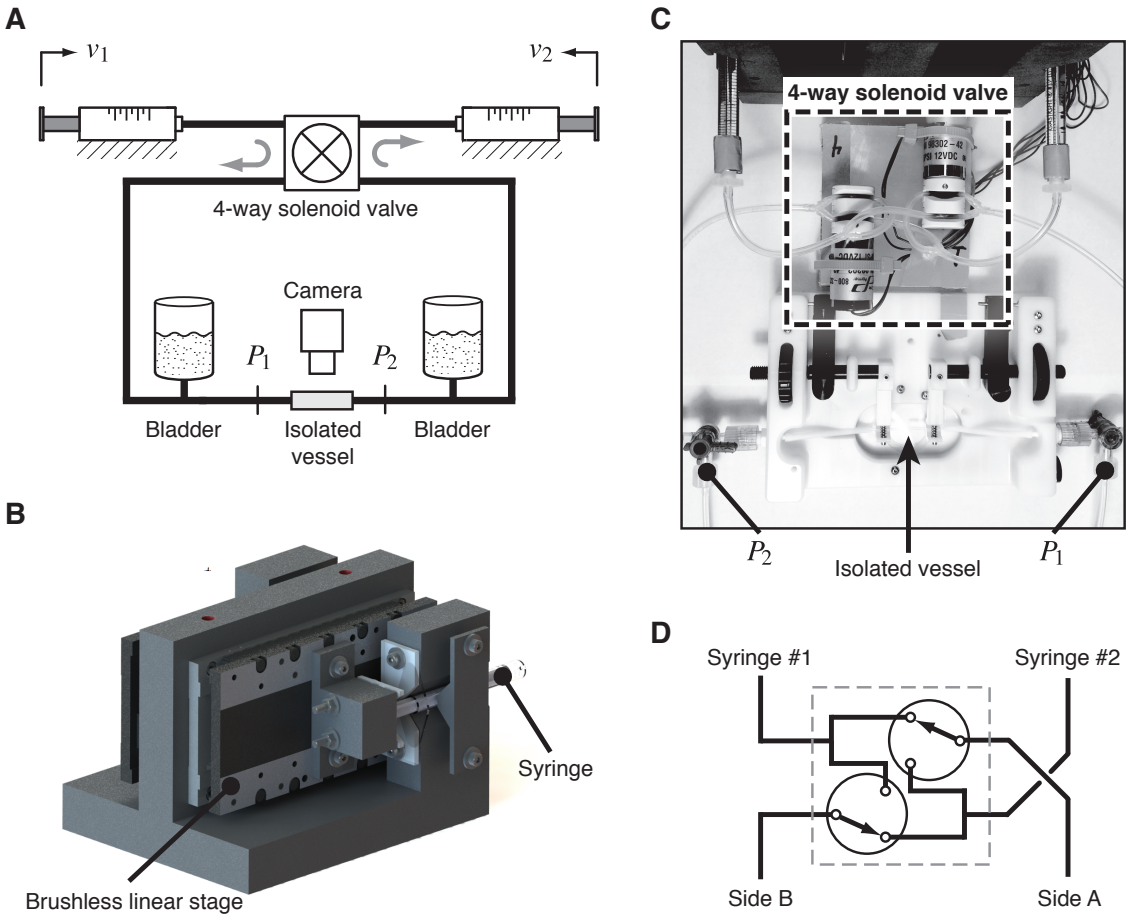


Figure 3.1: Ex-vivo lymphatic perfusion system (ELPS). (a) Syringes are actuated by two linear stages to independently control both $\Delta P = P_1 - P_2$ and $P_{\text{avg}} = (P_1 + P_2)/2$, during which a camera is recording the vessel's contraction dynamics. (b) Custom-built mount for the brushless linear stages to actuate each syringe plunger. (c) Overhead photo showing a portion of the ELPS and highlighting the location of the four-way solenoid valve, location of measurements P_1 and P_2, and location of the isolated vessel. (d) High-level schematic of the 4-way solenoid valve, which is comprised of two 3/2 solenoid valves. In the setting shown, Syringe #1 is connected to Side A and Syringe #2 to Side B; when the valves are switched, Syringe #1 is connected to Side B and Syringe #2 to Side A.

addition, two small fluid bladders are connected on each side of the fluid line in order to increase system compliance, helping mitigate high-frequency disturbances such as solenoid valve switching. The extraction and cannulation technique of the thoracic ducts is similar to previous studies [18], where a ~ 1 cm segment (free of valves) is taken from a Sprague-Dawley rat and cannulated on two resistance-matched glass pipets (400-500 μm tip diameter). Both the system's working fluid and the vessel bath are a physiological salt solution (PSS) (in mM: 145.0 NaCl, 4.7 KCl, 2.0 $CaCl_2$, 1.17 $MgSO_4$, 1.2 NaH_2 PO_4, 5.0 dextrose, 2.0 sodium pyruvate, 0.02 EDTA, and 3.0 MOPS). However, other types of media may be manually substituted after cannulation of the vessel: the author has successfully disassembled the tubing and changed the working fluid to a calcium-free PSS post-experiment to observe the vessel's tone-free diameter (data not shown).

Before operation of the ELPS, the vessel is warmed to 38°C for at least 20 min at $P_{\text{avg}} \approx 3$ cmH_2O. After the vessel is equilibrated, a validation protocol is performed in order to ensure proper vessel functionality before experimentation. This protocol consists of 5 min of applied zero pressure gradient, followed by a steady pressure gradient ($\Delta P = 1$ cmH_2O) that is also imposed for 5 min. Both of these conditions are imposed at an average transmural pressure, $P_{\text{avg}} = 3$ cmH_2O. In this way, contractile inhibition could be confirmed via the contraction frequency after the steady ΔP application like in previous studies [19, 18]. Contraction frequency for both the no-flow and steady ΔP condition were compared for 5 thoracic ducts using a paired two-tailed t-test. Upon the start of each experiment, the diameter tracing is recorded in real-time via a custom LabView program using data from a bright-field camera capturing at 30 fps as in other studies [18]. Both diameter data and other sensor data are synchronized post-experiment using recorded timestamps.

The digital electronic hardware for the compensator consists of a Microchip 32-bit PIC32 microcontroller running at 80 MHz, providing adequate computational headroom for the control loop to run reliably at 300 Hz. Specifically, the chipKIT Uno32 development board is used (Digilent, Pullman, WA), which is based on the Arduino Uno [8, 33].

The chipKIT Uno32 receives tracking commands serially at a baud rate of 921 kbps from a PC running a custom Python script [Appendix B.3], and it interfaces with the servo drives using a custom daughter board primarily consisting of two 32-bit LS7366R quadrature decoders for position measurement (LSI Computer Systems, Melville, NY) and a 16-bit, dual-channel AD5752 digital-to-analog converter (DAC) for the servo drive actuating signals (Analog Devices, Norwood, MA) [Appendix B.2]. In addition, the daughter board is connected to two 12-bit HSC 001PD digital pressure sensors for measuring both P_1 and P_2 (Honeywell, Golden Vally, MN). As with the Arduino, the software for this electronics platform uses high-level C++ functions to interface with basic hardware I/O, and in addition, the author has developed custom C++ drivers to easily interact with the components on the daughter board [Appendix B.4]. Detailed schematics and source code are also published on GitHub [32].

3.3.2 Control Scheme

The system may be described through the linear, discrete-time state-space equations,

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{G}\mathbf{x}_k + \mathbf{H}\mathbf{u}_k \\ \mathbf{y}_k &= \begin{bmatrix} \Delta P \\ P_{\text{avg}} \end{bmatrix}_k = \mathbf{C}\mathbf{x}_k \end{aligned} \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^{n\times 1}$ is the state vector, $\mathbf{G} \in \mathbb{R}^{n\times n}$ is the state matrix, and $\mathbf{u} \in \mathbb{R}^{2\times 1}$ is the input vector (voltages to each servo drive). The output vector, $\mathbf{y} \in \mathbb{R}^{2\times 1}$, represents the system output vector to be controlled, which is the transaxial pressure gradient, ΔP, and the average transmural pressure, P_{avg}. Due to the low pressures and pressure gradients this device produces, all nonlinear effects are ignored except for the saturation limit of the servo drives (± 10 V). The matrices $\mathbf{G}$, $\mathbf{H}$, and $\mathbf{C}$ are all determined through standard linear identification techniques; specifically, the author used MATLAB's `ssest()` function, which initializes parameter estimates using a noniterative subspace technique before subsequently refining

them using a prediction error minimization approach. It is noted that when using a black-box identification scheme such as this, the physical interpretation of the state variables, **x**, remains unknown since the state-space equations, shown in Eq. (3.1), have an infinite number of mathematical representations. In the case of this system, good agreement is achieved when $n = 6$ for a random binary input signal (± 0.6 V) bandwidth-limited to 30 Hz and sampled at 300 Hz [Fig. 3.2a], which is also the sampling time for control as mentioned previously. This sampling time was chosen both due to its reliable operation on the microcontroller used and its ability to capture all relevant dynamics expected and confirmed to be seen in the ELPS. The singular value plot of the ELPS is shown in Fig. 3.2b. In short, this plot displays the minimum and maximum singular values of the ELPS, which form an outer region bounding the system's frequency response (*i.e.* gain attenuation at each frequency).

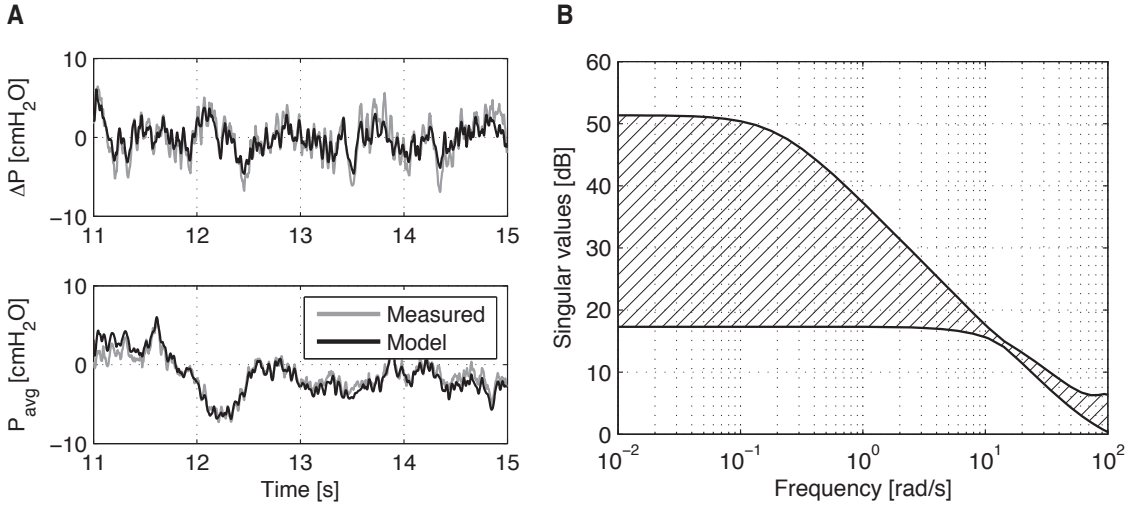


Figure 3.2: (a) ELPS validation data for the system identification using a rat thoracic duct. The input is a random binary signal (± 0.6 V) with a frequency band of 0-30 Hz generated with MATLAB. (b) Singular value plot indicating the outer region bounding the frequency response of the ELPS.

The model order chosen achieved good agreement with the validation data due to the inherent system dynamics of the setup. Since the primary components consist of two linear stages (each a second-order system) and two air bladders (with each compliance element

adding an additional system order), the author expected that the ELPS would be well-described by a sixth order system ($n = 6$). Indeed, upon testing various model orders during the identification process, the log of the Hankel singular values (which are the singular values of the product of the controllability and observability Gramians and represent how much each state contributes to the output characteristics of the model) shown in Fig. 3.3 confirm that a sixth-order system contributes to a majority of the ELPS's system dynamics. Although one might predict that the input-output characteristics of each linear stage (from the internal velocity control using voltage inputs) might possibly make each linear stage a third-order system, the author confirmed through a previous characterization of each linear stage that this additional pole is negligible since it is an order of magnitude faster than the real component of the next-fastest system pole.

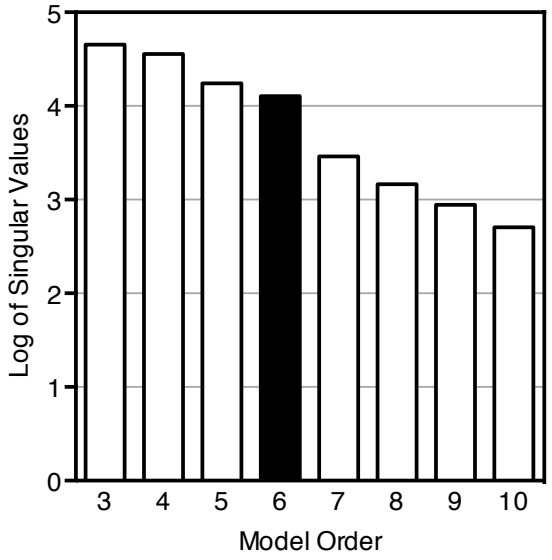


Figure 3.3: Log of the Hankel singular values, which represent the contribution of each state to the output characteristics of the model, for various model orders. Since it was predicted that $n = 6$ and the largest relative difference between model orders occurs after this order, the model may be assumed to be sixth-order.

Before a controller may be designed, the obtained model must be confirmed to be stable.

Thus, each eigenvalue, λ_i, of the state matrix, $\mathbf{G}$, is checked:

$$\boldsymbol{\lambda} = \begin{bmatrix} 0.8496 \pm 0.3138\,j \\ 0.8819 \pm 0.3137\,j \\ 0.9996 \\ 0.9698 \end{bmatrix} \tag{3.2}$$

Since $\|\lambda_i\| < 1$ in Eq. (3.2), stability is confirmed as the eigenvalues lie within the unit circle in the z-plane. In addition, before proceeding with designing the following state-feedback controller, both the observability, controllability, and output controllability of the system must be checked. To ensure observability, the rank of the observability Gramian, $\mathscr{O}$, must be equal to the order of the system:

$$\operatorname{rank} \mathscr{O} = \operatorname{rank} \begin{bmatrix} \mathbf{C} \\ \hdashline \mathbf{CG} \\ \hdashline \vdots \\ \hdashline \mathbf{CG}^5 \end{bmatrix} = 6 \tag{3.3}$$

Hence, the system is observable. Similarly, to ensure controllability, the rank of the controllability Gramian, $\mathscr{C}$, must be equal to the order of the system:

$$\operatorname{rank} \mathscr{C} = \operatorname{rank} \left[\mathbf{H} \,\vdots\, \mathbf{GH} \,\vdots\, \cdots \,\vdots\, \mathbf{G}^5\mathbf{H} \right] = 6 \tag{3.4}$$

Thus, the system is controllable. Lastly, because the output matrix, $\mathbf{C}$ is of full rank and the system is controllable, the system is also output controllable. As such, a state-feedback controller may now be designed.

The overall control objective of the system is to independently track the desired input waveforms, $\mathbf{y}_d$, (known *a priori*) as closely as possible while considering the input limitations of the actuators. Mathematically, we can describe this objective as minimizing a suitable quadratic cost function over some predictive time horizon, H_p:

$$J = \sum_{i=1}^{H_p} \mathbf{e}_{k+i}^{\mathsf{T}} \mathbf{Q}_i \, \mathbf{e}_{k+i} + \mathbf{u}_{k+i-1}^{\mathsf{T}} \mathbf{R}_i \, \mathbf{u}_{k+i-1} \tag{3.5}$$

where the tracking error vector, $\mathbf{e}$, is the difference between the desired output vector, $\mathbf{y}_d$, and the estimated output vector, $\tilde{\mathbf{y}}$; while the matrices $\mathbf{Q}_i$ and $\mathbf{R}_i$ are weighting matrices for

both the tracking error and the input, respectively, at the i^{th} prediction step. These weighting matrices are chosen by the designer and, in a sense, effectively adjust the emphasis of the minimization between the tracking error (to enhance performance) and the velocity of the syringes (to increase stability). This particular control scheme is, in general, known as model predictive control (MPC), and has been used for decades in industrial settings where multi-input, multi-output (MIMO) systems are prevalent [53, 17, 43]. Defining the following vectors,

$$\mathbf{Y}_d = \begin{bmatrix} \mathbf{y}_{d,k+1} \\ \hdashline \mathbf{y}_{d,k+2} \\ \hdashline \vdots \\ \hdashline \mathbf{y}_{d,k+H_p} \end{bmatrix}, \quad \tilde{\mathbf{Y}} = \begin{bmatrix} \tilde{\mathbf{y}}_{k+1} \\ \hdashline \tilde{\mathbf{y}}_{k+2} \\ \hdashline \vdots \\ \hdashline \tilde{\mathbf{y}}_{k+H_p} \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} \mathbf{u}_k \\ \hdashline \mathbf{u}_{k+1} \\ \hdashline \vdots \\ \hdashline \mathbf{u}_{k+H_p-1} \end{bmatrix} \tag{3.6}$$

the cost function in Eq. (3.5) may be recast in matrix/vector form:

$$J = (\mathbf{Y}_d - \tilde{\mathbf{Y}})^{\mathsf{T}} \mathbf{Q} (\mathbf{Y}_d - \tilde{\mathbf{Y}}) + \mathbf{U}^{\mathsf{T}} \mathbf{R} \mathbf{U} \tag{3.7}$$

where the weighting matrices $\mathbf{Q} = \text{diag}\left(\mathbf{Q}_1, \ldots, \mathbf{Q}_{H_p}\right)$ and $\mathbf{R} = \text{diag}\left(\mathbf{R}_1, \ldots, \mathbf{R}_{H_p}\right)$.

Rewriting Eq. (3.7) in terms of the estimated state, $\tilde{\mathbf{y}} = \mathbf{C}\tilde{\mathbf{x}}$, such that $J = J(\tilde{\mathbf{x}}, \mathbf{u})$, the stationarity condition, $\partial J / \partial \mathbf{U} = \mathbf{0}^{\mathsf{T}}$, may then be used to solve for the optimal input vectors, $\mathbf{U}^*$. Consequently, the control law governing these input vectors minimizes the quadratic cost function in Eq. (3.7). After a bit of algebra, this optimal control law may be written explicitly as

$$\begin{aligned} \mathbf{U}^* &= \left(\mathbf{K}_{CGH}^{\mathsf{T}} \mathbf{Q} \mathbf{K}_{CGH} + \mathbf{R}\right)^{-1} \mathbf{K}_{CGH}^{\mathsf{T}} \mathbf{Q} \left(\mathbf{Y}_d - \mathbf{K}_{CG} \tilde{\mathbf{x}}_k\right) \\ &\equiv \mathbf{K} \left(\mathbf{Y}_d - \mathbf{K}_{CG} \tilde{\mathbf{x}}_k\right) \end{aligned} \tag{3.8}$$

where the gain matrices are defined as follows:

$$\mathbf{K}_{CG} = \begin{bmatrix} \mathbf{CG} \\ \hdashline \mathbf{CG}^2 \\ \hdashline \vdots \\ \hdashline \mathbf{CG}^{H_p} \end{bmatrix}, \quad \mathbf{K}_{CGH} = \left[\begin{array}{c:c:c:c:c} \mathbf{CH} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hdashline \mathbf{CGH} & \mathbf{CH} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hdashline \mathbf{CG}^2\mathbf{H} & \mathbf{CGH} & \mathbf{CH} & \mathbf{0} & \mathbf{0} \\ \hdashline \vdots & \vdots & \vdots & \ddots & \mathbf{0} \\ \hdashline \mathbf{CG}^{H_p-1}\mathbf{H} & \mathbf{CG}^{H_p-2}\mathbf{H} & \cdots & \mathbf{CGH} & \mathbf{CH} \end{array}\right] \tag{3.9}$$

In addition, to obtain the input for just the current time, t_k, the control law in Eq. (3.8) may be written as

$$\mathbf{u}_k^* = \mathbf{K}_1 \left(\mathbf{Y}_d - \mathbf{K}_{CG} \tilde{\mathbf{x}}_k \right) \tag{3.10}$$

where $\mathbf{K}_1$ represents the first two rows of $\mathbf{K}$ (since $\mathbf{u} \in \mathbb{R}^{2\times 1}$). To obtain the estimated state, $\tilde{\mathbf{x}}_k$, a standard Luenberger observer is employed, which uses the system model to both predict the state vector at the next time step while also correcting for the error between the actual and predicted output at the current time step,

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{G}\tilde{\mathbf{x}}_k + \mathbf{H}\mathbf{u}_k + \mathbf{L}\left(\mathbf{y}_k - \tilde{\mathbf{y}}_k \right) \tag{3.11}$$

where $\mathbf{L}$ is a Kalman gain. Thus, the observer not only estimates the state, $\tilde{\mathbf{x}}$, but it also acts to minimize measurement noise (hence termed a Kalman filter).

The Kalmain gain, $\mathbf{L}$, which was calculated using MATLAB's `kalman()` function, uses both a user-defined process error covariance matrix, $\tilde{\mathbf{Q}} \in \mathbb{R}^{2\times 2}$, and measurement error covariance matrix, $\tilde{\mathbf{R}} \in \mathbb{R}^{2\times 2}$. The measurement error covariance matrix was estimated empirically by calculating the variance, σ_w^2, of the differential pressure sensors' resting noise (while reading a zero value):

$$\tilde{\mathbf{R}} = \begin{bmatrix} \sigma_w^2 & 0 \\ 0 & \sigma_w^2 \end{bmatrix}; \quad \sigma_n^2 = 0.1413 \text{ cmH}_2\text{O} \tag{3.12}$$

The process error covariance matrix, $\tilde{\mathbf{Q}}$, which is much more difficult to obtain, was assumed to be the identity matrix, $\mathbf{I}_{2\times 2}$. This matrix was chosen to be larger in order to increase the overall stability of the controller with respect to the measured output noise. All together, the control diagram is summarized in Fig. 3.4, and the MATLAB script used to generate the corresponding gain matrices in Eqs. (3.9)–(3.11) [Appendix B.5] is also provided on GitHub [32].

3.3.3 Post-Experiment Shear Stress Estimation

Due to the utilization of precision syringes and position encoders in the ELPS (which have a 0.5 μm resolution), it is possible to estimate the average flow rate through the vessel

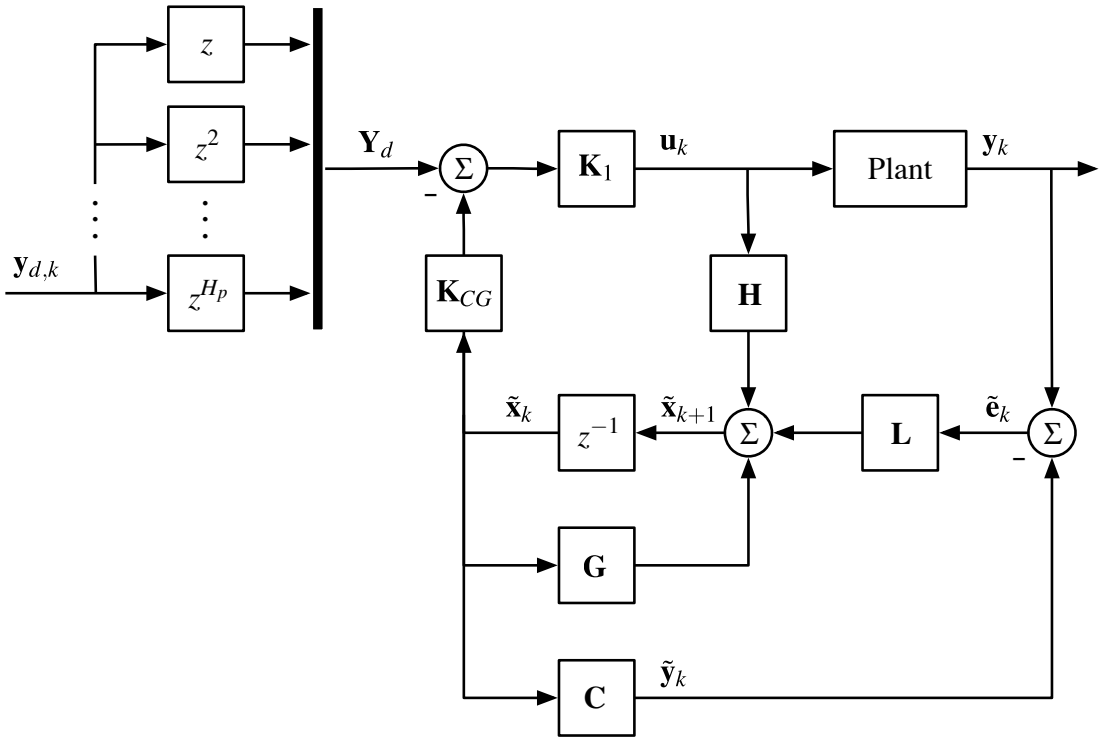


Figure 3.4: Explicit model predictive control (MPC) scheme with estimator used to compensate the plant, which consists of both the system in Fig. 3.1a and the servo drives.

over a certain window of time, Δt. Disregarding the transient dynamics by making this time window long ($\Delta t = 5$ sec used here; see Section 3.6), the average volume of fluid displaced out of one syringe and into the other during this time window should accurately approximate the flow rate through the vessel imposed by the system,

$$\bar{Q} \approx A_c \frac{1}{2\Delta t} \left[\sum_k \Delta x_{1,k} \delta_k - \sum_k \Delta x_{2,k} \delta_k \right]_{\Delta t} \tag{3.13}$$

where A_c is the cross-sectional area of the syringe plunger, $\Delta x_{i,k} = x_{i,k} - x_{i,k-1}$ is the incremental position change (backwards difference) of the i^{th} syringe plunger at the k^{th} time sample, and

$$\delta_k = \begin{cases} 1, & \text{Solenoid OFF} \\ -1, & \text{Solenoid ON} \end{cases} \tag{3.14}$$

Using this information, along with the mean diameter of the vessel, $\bar{D}$, over the same time window, the author calculated the estimated wall shear stress imposed on the vessel over time window Δt,

$$\bar{\tau}_w = \frac{32\mu\bar{Q}}{\pi\bar{D}^3} \tag{3.15}$$

where μ is the dynamic viscosity of the fluid, taken to be that of water at 38°C. It is noted that this wall shear stress estimation assumes Poiseuille flow, which has been shown to be a sufficient constitutive relation for describing lymphatic fluid flow [12, 11, 52] due to the fact that both the Reynolds number and Womersley number are very low. Hence, one may calculate both $\bar{Q}$ and $\bar{\tau}_w$ every Δt units of time to study slowly changing phenomena during an experiment [Fig. 3.9].

3.4 Results

As mentioned previously, the primary control objective of the ELPS is to independently track $\Delta P(k)$ and $P_{\text{avg}}(k)$ sent serially by the experimenter's PC to the compensator. Figure 3.5 demonstrates this dynamic tracking performance of the ELPS. As shown, various waveform combinations are possible, including both a varying ΔP with a constant P_{avg} and

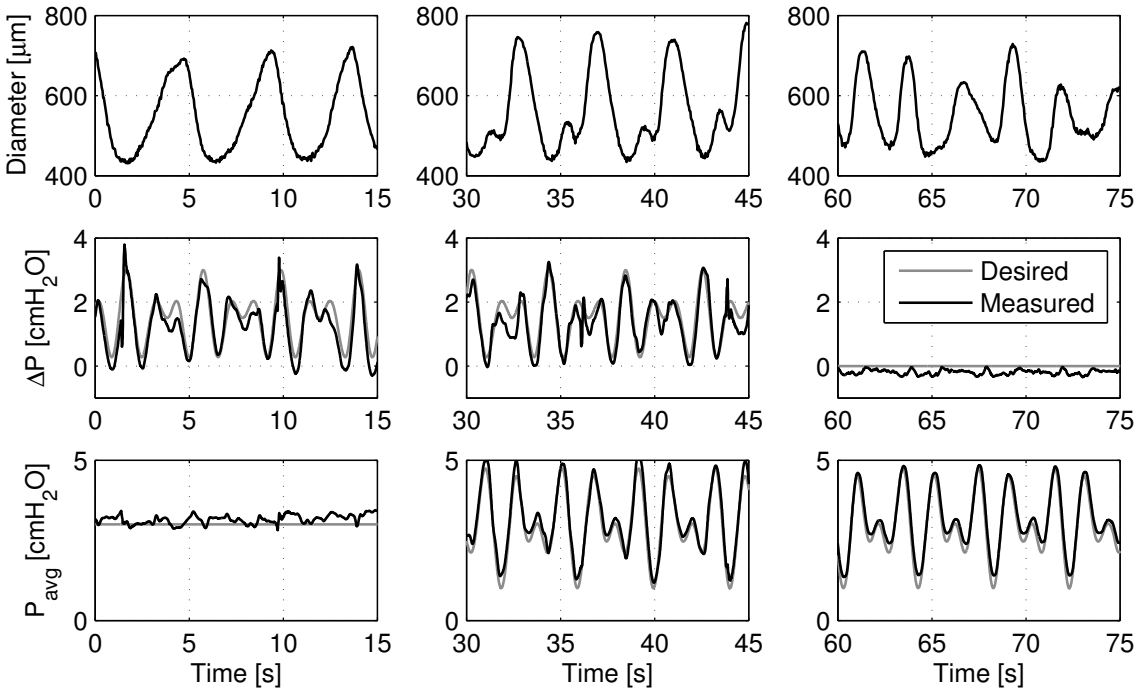


Figure 3.5: Various waveform combinations demonstrating ELPS tracking capabilities for both ΔP and P_{avg}. From left to right: time-varying ΔP with a constant P_{avg}, time-varying ΔP and P_{avg}, and a constant ΔP with a time-varying P_{avg}.

a varying P_{avg} with a constant ΔP. Additionally, the ELPS can independently and concurrently track different time-varying ΔP and P_{avg}. In each of these cases, these pressure waveforms were applied on an actively pumping rat thoracic duct approximately 600 μm in diameter. However, other vessel types are possible: Figure 3.6 shows the same waveform combinations being imposed on a much smaller vessel ($\sim$ 125 μm) from the rat mesentery.

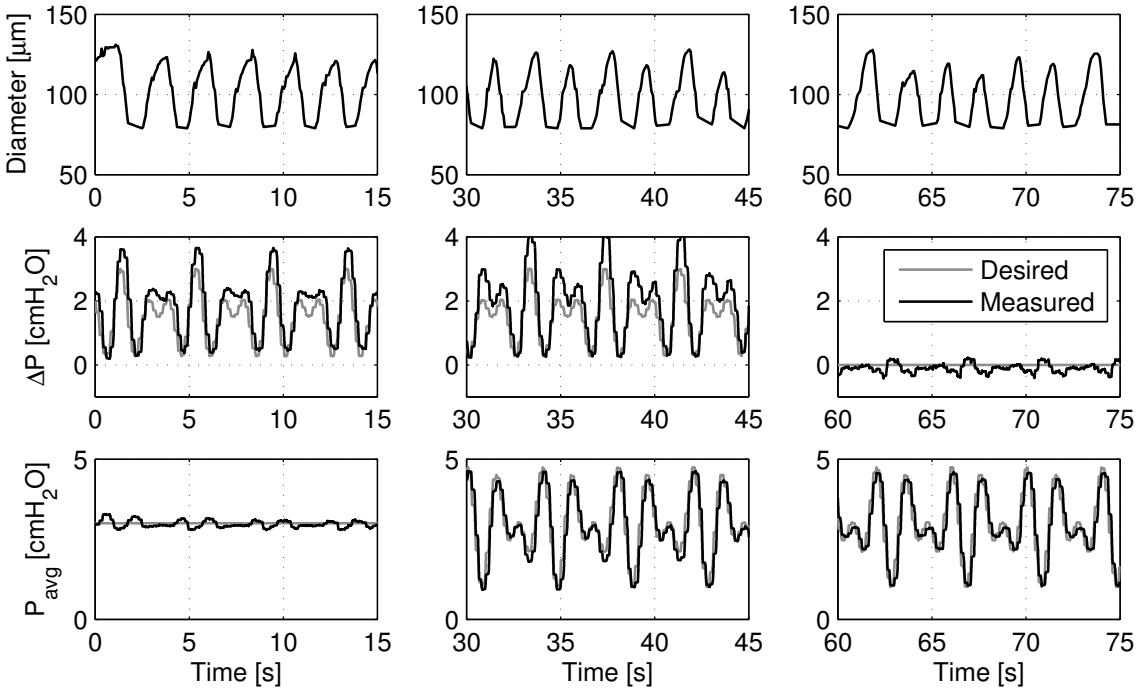


Figure 3.6: Various waveform combinations demonstrating ELPS tracking capabilities on a pumping vessel from the rat mesentery for different combinations of ΔP and P_{avg} (as in Fig. 3.5).

A more complete picture of the ELPS in operation may be seen in Fig. 3.7, where the ELPS simultaneously applied a constant P_{avg} (held at 3 cmH_2O) and an oscillatory ΔP based on a measured *in vivo* lymphatic waveform containing flow reversal. Specifically, the desired ΔP is a periodic function containing the first three harmonics of an oscillatory flow rate measured in the rat mesentery [30]. The corresponding diameter of the isolated rat thoracic duct is also shown along with the switching state of the solenoid valves, which is indicative of the volume of fluid being moved through the vessel. It is noted that the

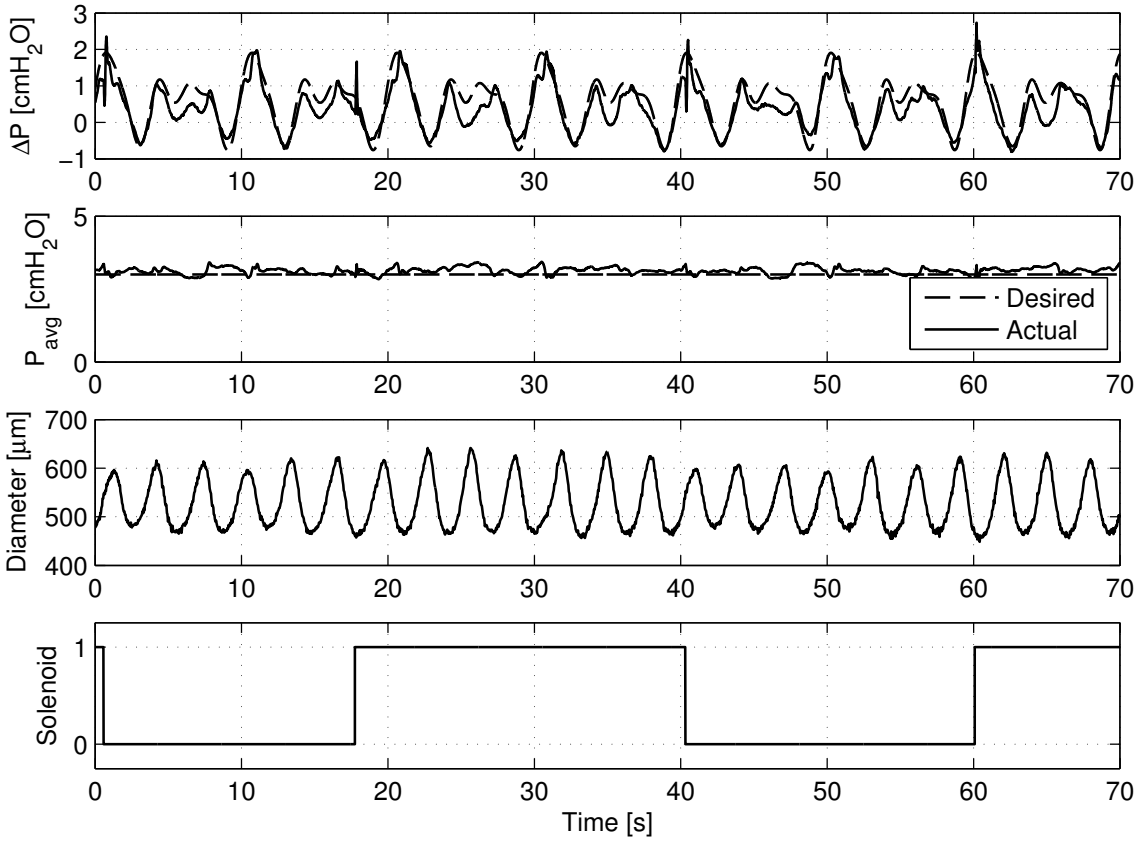


Figure 3.7: Varying ΔP based on the first three harmonics of a measured *in vivo* waveform from a rat [30], while P_{avg} remains constant. The corresponding thoracic duct diameter and solenoid state are also shown.

sharp spikes observed in the measured ΔP corresponds to these solenoid switching events, while the effect is less pronounced in the measured P_{avg}. Additionally, the author has not observed any abnormal contractile responses of the vessel during solenoid valve switching during a validation protocol performed prior to experimentation [Fig. 3.8]. Because these switching events are brief (resulting ΔP modulations are typically < 0.25 sec) and have a relatively small magnitude, the author believes they have negligible effect on normal lymphatic contractile function. To validate that the solenoid switching had no adverse effect on lymphatic pump function, the author confirmed a significant reduction ($p < 0.05$) in contraction frequency upon the onset of a steady ΔP (1 cmH_2O) during the validation protocol in 5 separate vessels [Fig. 3.8b].

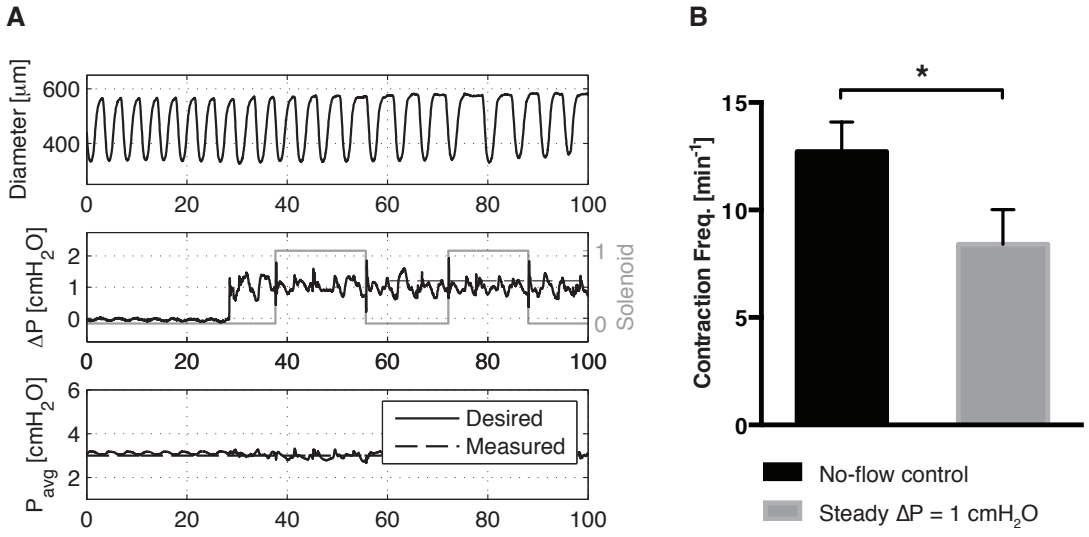


Figure 3.8: To validate proper vessel functionality before experimentation, a steady pressure gradient ($\Delta P = 1$ cmH_2O) is imposed for 5 min after a no-flow condition ($\Delta P = 0$), both at $P_{\text{avg}} = 3$ cmH_2O. (a) Despite switching events present after the onset of a steady ΔP, the resulting contraction profile does not display any apparent irregularities. (b) As expected from previous studies [19, 18], application of a steady ΔP results in a significant reduction in contraction frequency compared to a no-flow control ($p < 0.05$, $n = 5$). Error bars represent SEM.

However, although the number of solenoid switching events hints at the amount of

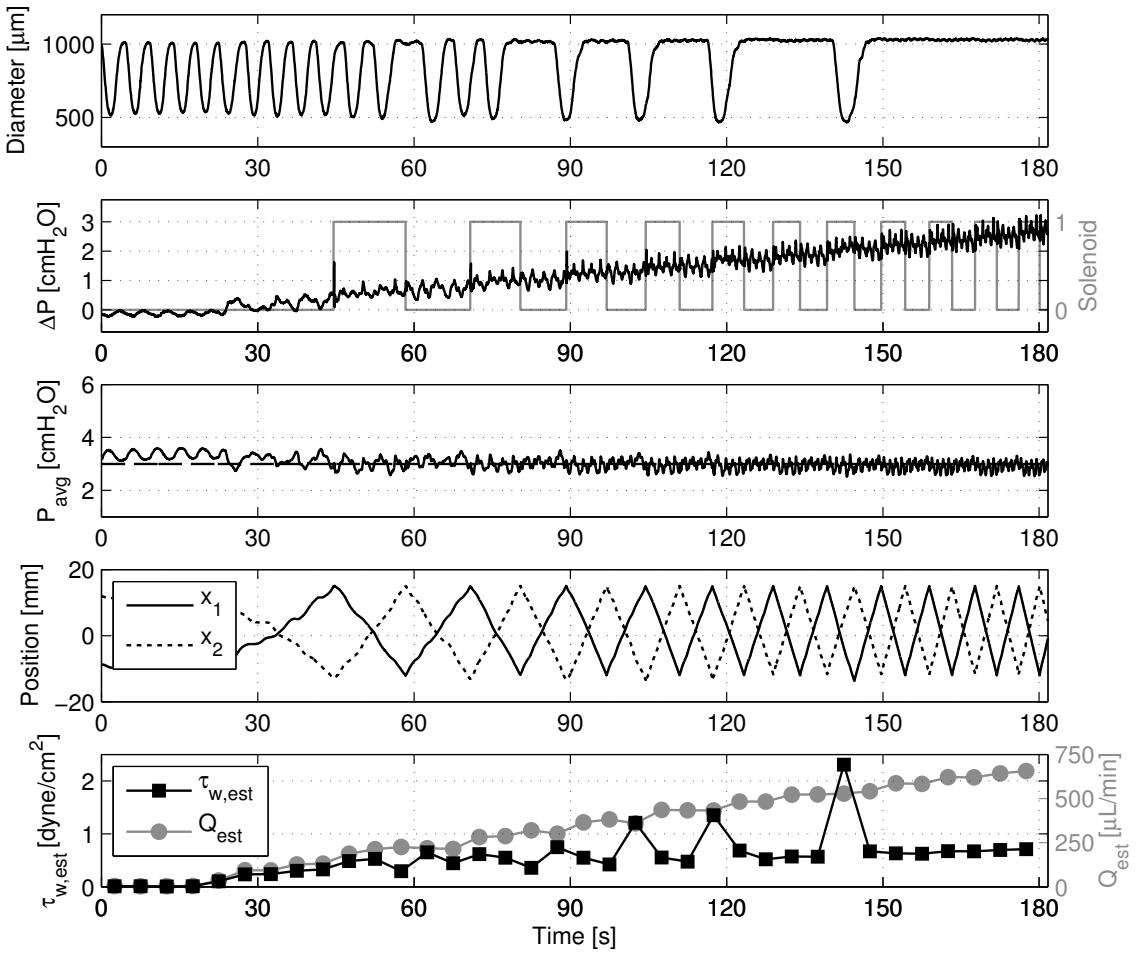


Figure 3.9: Example showing estimated shear stress, $\bar{\tau}_w$, calculated from estimated flow rate, $\bar{Q}$, and mean diameter, $\bar{D}$, over sequential time windows, $\Delta t = 5$ sec each, for a ramped ΔP. Transmural pressure remains constant at 3 cmH_2O, and the corresponding linear stage position, x_1 and x_2, are shown along with the solenoid state to help demonstrate the estimation procedure described in Section 3.3.3.

fluid being transferred through the vessel, this information may be combined with position encoder measurements for the stages to estimate the flow rate over multiple, 5 second windows (detailed in Section 3.3.3). Thus, with the flow rate estimation and diameter information in hand, the imposed wall shear stress may be estimated as well. An example of this approximation may be seen in Fig. 3.9, where ΔP is ramped from 0 to 3 cmH_2O over 3 min while holding P_{avg} constant at 3 cmH_2O. For this experiment, the instantaneous Poiseuille flow assumption for calculating shear stress should hold as both the Reynolds number and Womersley number are always less than 20 and 0.8, respectively. As ΔP increases, the rate of solenoid switching also climbs, resulting in a rising $\bar{Q}$. Nonetheless, as the flow rate increases, the thoracic duct's contraction is inhibited, which leads to a higher mean diameter, $\bar{D}$. The overall effect may be seen through $\bar{\tau}_w$—even as the flow rate continues to increase linearly, the estimated shear stress, $\bar{\tau}_w$, eventually levels off [Fig. 3.10]. In addition to observing dynamic trends, various functional metrics may also be calculated (as defined in many other lymphatic studies [19, 18, 11]) using the diameter data from the experiment. Table 3.1 shows several of these commonly used metrics for the diameter data in Fig. 3.9.

Table 3.1: Various metrics calculated from the diameter data in Fig. 3.9.

Metric	Value
Mean Diastolic Diameter [μm]	1019.5
Mean Systolic Diameter [μm]	502.0
Mean Contraction Frequency [min^{-1}]	6.27
Fractional Pump Flow [min^{-1}]	4.75
Mean RMS Velocity [$\mu m/s$]	147.1

3.5 Discussion

In order to impose arbitrary ΔP and P_{avg} waveforms on an isolated lymphatic vessel, the author chose one of the simplest mechanical configurations: two opposing and independently-controlled pistons (syringes) located on each side of the vessel. In this way, the resulting

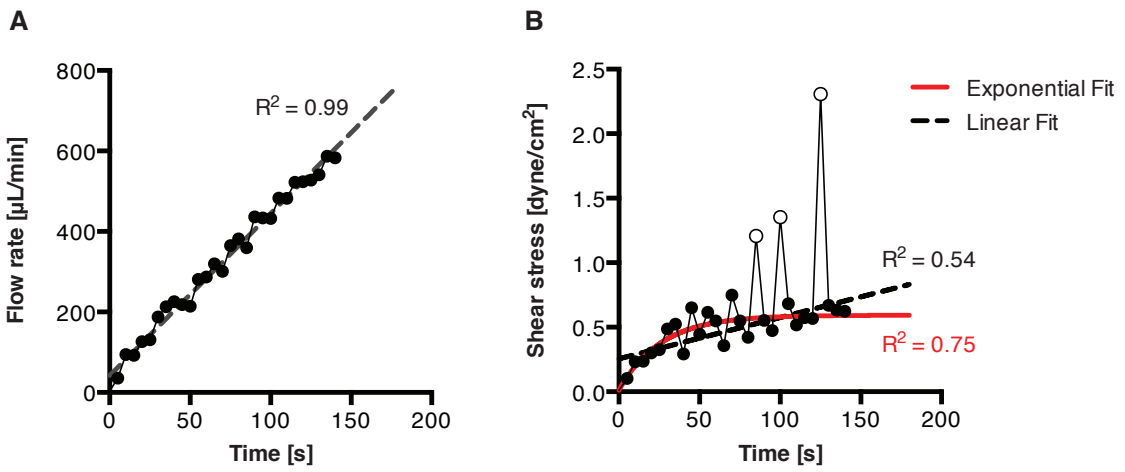


Figure 3.10: Estimated fits for the flow and shear stress profiles seen in Fig. 3.9. (a) The flow rate is confirmed to be linear, with $R^2 = 0.99$ for a linear fit. (b) However, the shear stress profile, which contains more point-to-point deviations due to the contractile activity of the vessel, is better fitted to an exponential function ($R^2 = 0.75$) than a linear function ($R^2 = 0.54$) since the shear stress begins to level off over time. To better visualize this trend, the three hollow points (corresponding to the last three contractions in Fig. 3.9 before complete flow inhibition) were excluded from both the linear and exponential fits.

system intrinsically has two dynamic modes—one which dictates ΔP when the syringes travel together, and one which dictates P_{avg} when the syringes operate in contrast. Additionally, the positive-displacement nature of the syringes, when coupled with the fast-moving MX80L brushless linear stages (which have a rate-limiting acceleration of 4 g's), allows for quick alterations of both ΔP and P_{avg}. Careful thought was given to minimize the tubing length where possible in order to reduce the effects of fluid inertance; however, small fluid-air bladders composed of larger tubing where necessary to increase system compliance to a level where disturbances (such as solenoid switches) were also minimized. Thus, trade-offs had to be made in the system configuration between dynamic performance and robustness of operation: the effects of this trade-off may be seen in Fig. 3.2b, where significant gain attenuation begins to occur between 1 and 10 rad/s. To validate that P_{avg} (even as a scalar value) may be used to approximate the pressure distribution in the vessel during flow, the author used $\bar{Q}$ to calculate that less that 1% of the pressure-drop between P_1 and P_2 occurs

within the vessel itself (assuming Poiseuille flow). This result is due to the fact that the length of the vessel is very small compared with the length of the cannula pipettes.

System identification of the ELPS, as mentioned previously in Section 3.3.2, consisted merely of finding a suitable linear state-space model in the discrete-time domain. Although good agreement is seen between the model and the validation identification data [Fig. 3.2a], remaining discrepancies may be attributed to nonlinearities in the setup. Considering the relatively low pressures and flows imposed by the system, expansion of the tubing and air bladders is minimal and the Reynolds number is always small, suggesting mostly linear behavior for the system dynamics. However, static friction (stiction) inherent in the syringes and the electrodynamics of the linear motors themselves certainly contain nonlinearities. Indeed, the nonlinear cogging force present in the linear stages is most likely a major culprit considering this is a known limitation of direct-drive brushless motors [27].

It is important to note that the author chose to use a lymphatic vessel segment free of valves during the ELPS identification in order to minimize system nonlinearities. Since the presence of one-way valves found in the collecting lymphatics could introduce significant nonlinearities in the system, the resulting model's predictive nature would be greatly reduced. Likewise, as the presence of one-way valves would prevent the application of fluid flow in a direction opposite to the valves, the author deliberately chose to use valve-free vessel segments during experiments with the ELPS. In this way, the ELPS could impose transaxial pressure gradients (hence flow rates and corresponding wall shear stresses) of any directionality across the vessel's endothelium. Of course, since the system model used is linear and based on a valve-free vessel segment, tracking control of the ELPS could cause unreliable operation when used with vessels with valves while imposing ΔP directionality in contrast to the valve direction. This aspect of the ELPS is certainly a limitation and would require further study to verify system compatibility with lymphatic vessels that contain valves.

Nevertheless, tracking control of the ELPS is quite advanced for *ex vivo* perfusion systems due to its ability to independently control both ΔP and P_{avg}—even though they may be time-varying. This capability is important: independent tracking control affords the ultimate flexibility to perform a wide range of experiments, including single-factor studies that aim to isolate specific mechanically-mediated mechanisms. For instance, the ability to single out shear-induced mechanisms is particularly interesting in light of the complex (and mostly oscillatory) flow rates experienced by the lymphatics *in vivo* [11, 30], where the ability to provide a dynamically-varying ΔP while holding P_{avg} constant would be necessary. Using an explicit MPC control law simplified for linear systems [Section 3.3.2], the compensator is able to utilize the predictive power of the identification model along with the feedback capabilities of the Kalman filter to generate plant inputs that minimize the desired performance objective (cost function). In other words, the MPC control law is designed in such a way to minimize both the multi-output tracking error and the multi-input voltage levels, which in turn increases both system performance and robustness. Although MPC and its application is not new, nobody has utilized such an advanced MIMO control scheme for an *ex vivo* perfusion system to track multiple signals.

However, despite its advanced nature, the way in which MPC was implemented in this case provides a very structured and simplified way of achieving the desired control performance for the system. Unlike many MPC algorithms that are nonlinear and require iterative solutions between time steps, the designer may employ an explicit linear control law [Eq. (3.10)] and state estimator [Eq. (3.11)] since the ELPS is described well through the linear state-space equations [Eq. (3.1), Fig. 3.2a]. Not only does this simplify hardware implementation, but it also allows for less-advanced electronics hardware to be used (such as the microcontroller platform used in this study). In order to adjust the controller performance, the designer merely needs to alter the weighting matrices, $\mathbf{Q}$ and $\mathbf{R}$, based on desired emphasis between tracking performance and control robustness, and then recompute the gain matrix, $\mathbf{K}_1$. This systematic approach to calculating the control gains is

a major advantage compared to traditional controllers like PID, where finding the correct combination of gains to achieve comparable independent tracking performance would be extremely difficult (if not impossible) for MIMO systems.

Of course, the ELPS's control system does have limitations. For instance, because the identification model and MPC control law are purely linear, the predictive capabilities of the system are restricted due to inherent nonlinear behavior. Thus, dynamic control performance could be improved by using a nonlinear MPC algorithm that takes various nonlinearities—such as the brushless linear stages' cogging forces—into account. In addition, more advanced nonlinear control algorithms could also be used: sliding mode control, which is known for excellent disturbance rejection, could be a promising candidate to negate the disturbances caused by both the linear motors and solenoids. Nevertheless, despite these dynamic control limitations, the ELPS is certainly able to produce ΔP and P_{avg} waveforms within the frequency range of lymphatics, which is typically less than 1 Hz [24, 11, 9]. Naturally, more straight-forward modifications, such as the addition of integral control, could also improve the tracking performance by eliminating steady-state error.

Another limitation of the ELPS is the inability to measure flow rate through the vessel in real time. Although recording this measurement at every sampling time would be desirable for estimating fluid shear stress, the flow rate through the vessel is very low (< 1 mL/min) and therefore difficult to measure with commercial flow sensors. Due to this limitation, the ELPS instead controls ΔP, which ultimately dictates the flow rate and is, therefore, commonly used in isolated lymphatic vessel studies [19, 18, 49]. Despite the inability to measure real-time flow rate, the author has demonstrated a technique to estimate the applied flow rate through the vessel *post hoc* (and thus wall shear stress) over multiple, 5 second windows due to the positive-displacement nature of the syringes. Although this is certainly a limitation as real-time flow rate (and therefore peak shear stress) information would be essential to studying short-term shear stress dynamics, these estimated values do have the ability indicate trends over a longer time scale. For instance, as ΔP (and hence $\bar{Q}$)

is increased linearly to an abnormally high magnitude in Fig. 3.9, the consistent pumping of the vessel is actively reduced—eventually resulting in complete pumping inhibition (as expected). The estimated parameter, $\bar{\tau}_w$, demonstrates this phenomenon clearly: as vessel contraction is inhibited (thus increasing $\bar{D}$), $\bar{\tau}_w$ begins to curtail despite the linearly increasing $\bar{Q}$ [Fig. 3.10]. This scenario could be representative of a normal physiological response of the collecting lymphatics to a large influx of lymph, where the lymphatic vessel would want to behave more as a conduit than a pump by reducing its overall resistance to the incoming fluid load.

In brief, the ELPS and its ability to independently control ΔP and P_{avg} waveforms make it unique among isolated lymphatic vessel systems and even state-of-the-art vascular *ex vivo* perfusion systems. As discussed, this capability is crucial for true single-factor studies that seek to isolate mechanically-mediated mechanisms specific to either fluid shear stress or circumferential stress. Furthermore, applying dynamic and differing ΔP and P_{avg} waveforms in concert could be helpful in teasing out potential interdependencies between mechanisms affected by these two significant mechanical forces. In all, the ELPS is a promising platform for studying the functional role of mechanics on isolated lymphatic vessels, allowing for new studies and potentially leading to significant discoveries relating to both physiologic and pathophysiologic cases of lymphatic fluid transport.

3.6 Time Window Length Calculation

In order to determine what constitutes a long Δt, we must first estimate the transient dynamics relating the instantaneous syringe velocity (averaged between the two syringes), v_{avg}, to the transaxial pressure gradient, ΔP. The instantaneous syringe velocity averaged between the two syringes is defined as follows:

$$v_{\text{avg}}(k) = \frac{\Delta x_{1,k}\delta_k - \Delta x_{2,k}\delta_k}{2T_s} \tag{3.16}$$

where T_s is the sampling time of the identification (see Section 3.3.3 for additional nomenclature). Thus, using the data from the identification experiment shown in Fig. 3.2, one

may reconstruct another identification of a single-input, single-output (SISO) system with v_{avg} as the input and ΔP as the output. The validation data for this identification is shown in Fig. 3.11a (model order, $n = 4$, with good agreement), while the corresponding dynamic characteristics of this model is shown in Fig. 3.11b.

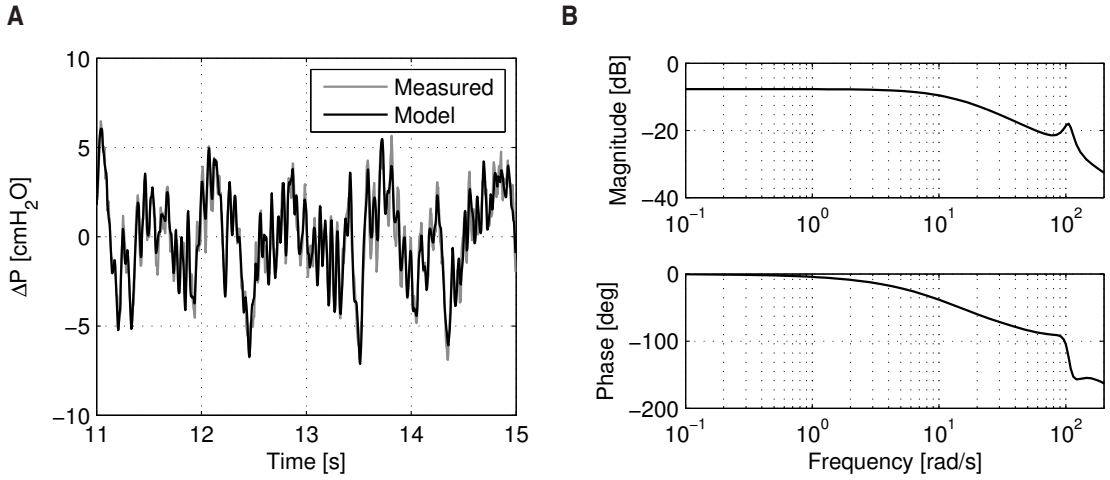


Figure 3.11: (a) Validation data (from the experiment in Fig. 3.2) for the SISO dynamic model of the ELPS with average instantaneous syringe velocity between the two syringes as the input and transaxial pressure gradient as the output. (b) Bode plot of this SISO model of the ELPS showing the frequency response.

Of course, this model does not take into account the dynamics between ΔP and the flow rate through the vessel, which certainly contain some fluid compliance and inertance. However, assuming these effects are on the same order of magnitude as between v_{avg} and ΔP (or smaller), a value of Δt much larger than these dynamics should suffice. To quantify the speed of these dynamics, the the 2% settling time, T_{set}, is found from simulating the model in Fig. 3.11 in response to a step input. For this model, T_{set} is approximately 0.3 sec; thus, to ensure Δt is long ($> 10\,T_{\text{set}}$), the author defines:

$$\begin{aligned} \Delta t &\gg T_{\text{set}} \\ &= 5 \text{ sec} \end{aligned} \tag{3.17}$$

CHAPTER IV

EFFECTS OF DYNAMIC SHEAR AND TRANSMURAL PRESSURE ON WALL SHEAR STRESS SENSITIVITY IN COLLECTING LYMPHATIC VESSELS

4.1 Abstract

Due to both the intrinsic pumping innate to the collecting lymphatics and other extrinsic factors, the lymphatic endothelium is exposed to a wide range of time-varying mechanical forces, specifically fluid shear stress and circumferential stress. Because both of these components have been shown to alter lymphatic contractility *ex vivo*, the dynamic characteristics of these loads, which are smaller and more diverse than those seen in the blood vasculature, could play an important role in normal lymphatic pump function. Hence, the author set out to investigate the affects of dynamic shear stress on collecting lymphatics, including (1) how the shear sensitivity is affected by transmural pressure; (2) if a dynamic transaxial pressure gradient (*i.e.* shear stress) suppresses the typical shear-induced inhibitory effect on contraction frequency; and (3) whether or not a dynamic shear stress can independently coordinate lymphatic contractile activity. Specifically, the author used a previously-developed *ex vivo* lymphatic perfusion system (ELPS) [Chapter 3] to independently vary these two forces, including a multitude of pressure gradient waveforms consisting of ramps and sine waves on a total of 9 rat thoracic ducts while controlling for transmural pressure and measuring diameter changes. In brief, the mean shear sensitivity at an average transmural pressure of 5 cmH_2O (1.23 dyne/cm^2) was significantly less than the shear sensitivity at an average transmural pressure of 3 cmH_2O (1.73 dyne/cm^2) during applied pressure gradient ramps. Normalized contraction frequency was not inhibited on vessels during imposed pressure gradient sine functions with respect to a steady-flow, even

though the mean wall shear stresses were not significantly different. Additionally, a pressure gradient sine function was shown to synchronize lymphatic contractions to the exact frequency of the waveform once applied. These results demonstrate for the first time that transmural pressure affects the shear sensitivity of the lymphatic endothelium, with a higher transmural pressure increasing shear stress sensitivity. Also, time-varying shear stress not only has the ability to obstruct inhibition of phasic contraction frequency typically seen *ex vivo*, but when large enough also has the ability to independently coordinate contractions. Thus, for the first time the author has provided concrete evidence that dynamic shear has the ability to independently influence contractile coordination and could play an important role in the normal contractile function of collecting lymphatic vessels.

4.2 Introduction

The contraction dynamics of collecting lymphatic vessels, which are categorized as either phasic or tonic and often referred to as the intrinsic pump; are complex, time-varying, and have been shown to be sensitive to how they are mechanically loaded [40, 19, 9]. Since lymph formation can vary widely even during normal, physiologic circumstances due to extrinsic factors [36] (such as skeletal muscle contraction, respiration, and interstitial fluid formation), it has been hypothesized that the lymphatics' sensitivity to the local mechanical environment aids in optimizing lymph transport [19]. This ability to sense and respond to local mechanical forces would certainly be beneficial to the vessels' intrinsic pumping, which must actively adapt to these rapidly varying loads *in vivo*. For instance, like the heart, collecting lymphatic vessels have been shown to quickly react to different levels of transmural pressure [40, 24] and preload/afterload [59, 10]—parameters which change continuously based on levels of lymph formation. Much like in the heart, the lymphangions' response to these loads is tuned in such a way to maximize fluid output. Furthermore, the rate of applied pressure load has also been shown to alter pump function [9]; which, since these rate-sensitive changes were shown to be different to that of the portal vein, suggests

that the lymphatics may have adapted to compensate for the rapidly varying changes in load that can occur *in vivo*. Considering that the lymphatics' sensitivity to mechanical loading has also been shown to differ based on anatomical location [18], the notion of vessels actively adapting to their mechanical environment to function optimally is further supported.

In addition to a contractile dependency on transmural pressure, isolated vessel studies have also demonstrated inhibition of lymphatic pumping amplitude, frequency, and tone in response to luminal fluid shear stress [19, 18]. In these studies, higher magnitudes of applied transaxial pressure gradient resulted in higher levels of inhibition for these contractile parameters. These responses are mediated by the lymphatic endothelium and have been shown to use similar mechanisms that regulate vasoactive responses in the blood vasculature [31]. Likewise, the transition of flow direction has also been shown to mediate lymphatic contractility. Specifically, Gashev *et al.* demonstrated that rapidly changing from orthograde to retrograde flow (as well as from orthograde or retrograde flow to no flow) temporarily increased contraction frequency in isolated lymphatic vessels [19]. They proposed that this fast chronotropic response could be a useful physiologic mechanism to prevent unneeded inhibition of lymphatic contractions during regular pumping activity. However, although lymphatic flow has been shown to be rapidly time-varying (and often oscillatory) *in vivo* [11], no one has yet to verify this fast chronotropic response in a collecting lymphatic exposed to a continuously-varying pressure gradient waveform (while also controlling for the effects of average transmural pressure).

Moreover, besides being attributed to assorted functional changes in collecting lymphatic vessels, shear stress has previously been implicated as a self-regulatory element in isolated lymphatic vessels [19, 20]. Hence, given that a continuously-varying pressure gradient may not inhibit contractile activity due to the unique flow waveforms typically imposed on the lymphatic endothelium seen *in vivo*, this dynamic shear stress could play an essential role to the contractile coordination observed among lymphangions [68]—a possibility which could greatly enhance lymph transport efficiency. Indeed, computational

studies have suggested that contractile coordination among chains of lymphangions with respect to their relative phase difference could greatly enhance overall lymph transport efficiency [2]. However, no one has demonstrated that the transaxial pressure gradient (hence fluid shear stress) is able to dynamically coordinate contractile activity independent of transmural pressure.

With respect to the magnitude of fluid shear stress in lymphatics, the levels of shear experienced by LECs *in vivo* are much less compared to similar-sized blood microvessels [24, 11]. This difference in sensitivity could be attributed in part to a marker such as vascular endothelial growth factor receptor-3 (VEGFR-3), a growth factor receptor expressed differentially on LECs; however, very little is known about the factors mediating the shear stress sensitivity of LECs. In addition to VEGFR-3, one potential candidate could be the transmural pressure exerted on the vessel. Given that isolated blood vessel studies have previously shown interplay between intraluminal pressure-induced responses with flow-induced dilations [34, 60], this coupling could be an important factor in normal contractile function given the enhanced mechanical sensitivity of lymphatics [39]. Additionally, blood endothelial cell responses to fluid shear stress are known to be partly mediated by junctional proteins (specifically, VE-cadherin, PECAM-1, and VEGFR-2) [63], with fluid shear stress recently being shown to increase the force on the junctional protein PECAM-1 [7]. All together, this evidence suggests that the transmural pressure within a collecting lymphatic, which certainly affects circumferential stress within the endothelium, and thus the force exerted on the junctional proteins, could assist in mediating the vessel's shear sensitivity.

Despite these questions regarding both the effects of dynamic shear stress on contractility and of transmural pressure on shear sensitivity, investigators have remained incapable of addressing them due to the lack of adequate experimental tools. For instance, because the flow rates present in the lymphatics are very low ($<$ 1 mL/min [11, 30]), measuring fluid shear stress is difficult in these isolated vessel preparations due to the lack of adequate commercial flow sensors. Additionally, no one has been able to construct an *ex vivo*

perfusion system capable of imposing independent transaxial pressure gradient and average transmural pressure waveforms to perform dynamic, single-factor shear stress studies. In this respect, the author has developed an *ex vivo* lymphatic perfusion system (ELPS) capable of both estimating fluid shear stress and independently controlling both transaxial pressure gradient and average transmural pressure on an isolated lymphatic vessel [Chapter 3]. In this current study, the author utilizes this system to (1) quantify shear sensitivity in the collecting lymphatics and investigate how it is affected by transmural pressure, (2) substantiate the idea that continuously-varying pressure gradient waveforms within a physiologic frequency range do not have an inhibitory effect on contraction frequency (controlling for average transmural pressure), and (3) determine whether or not a dynamic transaxial pressure gradient (*i.e.* shear stress) can independently coordinate lymphatic contractile activity.

4.3 Central Hypotheses

Given the motivating literature in Section 4.2, the author's central hypotheses are as follows:

- **Hypothesis 1:** The functional changes seen in the collecting lymphatics in response to fluid shear stress are inextricably linked to the functional changes caused by transmural pressure.

- **Hypothesis 2:** Contractile inhibition normally observed in collecting lymphatic vessels due to steady fluid shear stress can be obstructed when imposing a continuously-varying shear profile within a physiologic frequency range, independent of transmural pressure.

- **Hypothesis 3:** When peak amplitudes are large enough, dynamic shear stress has the ability to coordinate lymphatic contractions independently from effects caused by transmural pressure.

Given the three hypothesis listed, Section 4.4 will describe the materials and methodologies used to address these questions.

4.4 Materials and Methods

4.4.1 Experimental Hardware

As described in Chapter 3, the ex-vivo lymphatic perfusion system (ELPS) consists of two independently actuated glass syringes connected in a closed-loop fashion on each side of an isolated vessel preparation [Fig. 3.1]. The system is connected by 1/16" ID, 1/8" OD Tygon tubing with the exception of the small segments (silicon) placed in the two three-way pinch solenoid valves (Cole-Parmer, Vernon Hills, IL), which are joined in such a way to effectively form a four-way valve. In this way, when one syringe has expelled a majority of its fluid, the four-way solenoid valve switches while the syringes begin to propagate in the opposite direction. Thus, the system can maintain the directionality of flow with respect to the isolated vessel for an indefinite period of time [26]. Each syringe is 100 μL in volume (Hamilton Company USA, Reno, NV) and is actuated independently by a MX80L brushless linear stage powered by a ViX 250AH servo drive (Parker Hannifin Corp., Rohnert Park, CA), which permits the generation of independent pressure gradient ($\Delta P = P_1 - P_2$) and average transmural pressure ($P_{\text{avg}} = (P_1 + P_2)/2$) waveforms [Chapter 3, Fig. 3.5].

The controller hardware consists of a chipKIT Uno32 microcontroller development board (Digilent, Pullman, WA), which is based on the Arduino Uno software platform [8, 33], running the control loop at 300 Hz. Again, as mentioned in Chapter 3, the controller receives desired waveform values serially at a baud rate of 921 kbps from a PC running a custom Python script, and it connects to the servo drives using a custom circuit board consisting of two 32-bit LS7366R quadrature decoders for position measurement (LSI Computer Systems, Melville, NY) and a 16-bit, dual-channel AD5752 digital-to-analog converter for the servo drive actuation signals (Analog Devices, Norwood, MA).

In order to measure pressure, the circuit board is connected to two 12-bit HSC 001PD digital differential pressure sensors (Honeywell, Golden Valley, MN) measuring P_1 and P_2 with respect to atmosphere (located on each side of the cannula pipettes [Fig. 3.1]). Upon the start of an experiment, the diameter tracing was recorded in real-time via a custom LabView program using data from a bright-field camera capturing at 30 fps, as in other studies [18]. Both diameter data and other sensor data were synchronized post-experiment using recorded timestamps.

4.4.2 Animals and Isolated Vessel Preparation

The extraction and cannulation technique of the thoracic ducts was similar to previous studies [19, 18], where a $\sim$1-cm segment (free of valves) was taken from a Sprague-Dawley rat and cannulated on two resistance-matched glass pipets 350-500 μm in tip diameter. In total, the author examined the contractile activity of thoracic ducts from 9 Sprague-Dawley rats. Once exteriorized, each thoracic duct segment was cannulated in a warm (38°C) bath of physiological salt solution (PSS) (in mM: 145.0 NaCl, 4.7 KCl, 2.0 $CaCl_2$, 1.17 $MgSO_4$, 1.2 NaH_2 PO_4, 5.0 dextrose, 2.0 sodium pyruvate, 0.02 EDTA, and 3.0 MOPS) that was pH-adjusted to 7.4. After the cannulation procedure was completed, the cannulation chamber was carefully integrated into the ELPS tubing (already perfused with warm PSS) as to avoid forming bubbles then placed onto the microscope stage. Once positioned, each vessel was allowed to equilibrate at 38°C for 20 min at a transmural pressure of 3 cmH_2O while regular contractile activity was established. In order to confirm that the vessel preparation was functioning correctly, the author sequentially imposed an average transmural pressure of 1, 3, 5, and then 3 cmH_2O with no transaxial pressure gradient and then a transaxial pressure gradient of 1 and 3 cmH_2O with an average transmural pressure of 3 cmH_2O for 5 min each (30 min total). In addition to serving as control conditions for that vessel, the author could ensure that decreased contractile frequency, tone, and amplitude occurred during applied pressure gradients and that increased contractile frequency occurred during high transmural

pressures [19, 18]. At the end of the experiment, each vessel was equilibrated in Ca^{2+}-free PSS for 20 min and subsequently exposed to an average transmural pressure of 1, 3, and 5 cmH_2O to observe the resting diameter (for calculating tone) at each pressure.

4.4.3 Fluid Shear Stress Sensitivity

In order to estimate the fluid wall shear stress, τ_w, imposed on the vessel, the author employed the Poiseuille flow constitutive relation,

$$\tau_w = \frac{32\mu Q}{\pi D^3} \tag{4.1}$$

where μ is the dynamic viscosity of the working fluid (taken to be that of water at 38°C), Q is the flow rate through the vessel, and D is the diameter of the vessel. This relationship has been shown to be sufficient for describing lymphatic fluid flow both experimentally [11] and computationally [52] due to the fact that both the Reynolds number and Womersley number are very low. Because the diameter is recorded in real time using video from a 30-fps camera, the author needed only to estimate the flow rate, Q, through the vessel in order to determine the fluid wall shear stress, τ_w. Fortunately, due to the utilization of precision syringes and position encoders in the ELPS, it is possible to estimate the average flow rate through the vessel in sequential 5-sec time windows, ignoring potential transient dynamics within each window [Section 3.3.3]. In other words, the average volume of fluid displaced from one syringe into the other during each 5-sec time window should accurately approximate the flow rate through the vessel imposed by the system over that time.

With these values in hand, the author used the ELPS to impose a transaxial pressure gradient ramping from 0 to 3 cmH_2O over 3 min while simultaneously holding the average transmural pressure constant at both 3 and 5 cmH_2O. Each ramp was preceded by 5 min of a zero axial pressure gradient at that respective average transmural pressure to allow the vessel to equilibrate again and resume contraction. In this way, the flow rate through the vessel (and hence the shear stress imposed on the vessel) could be estimated *post hoc*, and the author could determine at which point contractile inhibition occurs through observation

of the vessel diameter. The author interpreted this information as the shear sensitivity of the vessel—in other words, the shear stress threshold at which the vessel begins to actively adjust its diameter based on the incoming fluid flow. However, to accomplish this, the author first had to define an objective metric to determine at what point this contractile inhibition begins to occur. Utilizing the linear relationship between τ_w and Q in the Poiseuille flow equation as shown in Eq. (4.1), a linearly increasing flow rate due to the ΔP ramp would also result in a linearly increasing wall shear stress assuming the average vessel contractile pattern (*i.e.* average vessel diameter) remains unchanged. Hence, deviations from this linear trend may be attributed to active behavioral modifications of the vessel through contractile amplitude, frequency, or tone—all of which affect the mean diameter of the vessel over time.

To determine the point at which point the estimated shear stress deviated from a linear trend, the author performed a least squares curve fit of the estimated shear stress values to a well-fitting function. In particular, the following function was used:

$$\tau_w = a(1 - e^{-t/T_c}) \tag{4.2}$$

where both a and T_c are the parameters determined during the fit, only using data after the flow rate is above some small threshold and until the last contraction is observed [Fig. 4.2]. In this way, the fit is only performed on estimated shear stress values occurring during the time where the flow rate is nonzero and before complete contractile inhibition occurs. The value, T_c, is known as the time constant in dynamics theory and is commonly used to describe systems that can be represented by Eq. (4.2). Thus, the author used this information to determine the shear sensitivity of the vessel: the wall shear stress value at T_c sec after positive flow occurred (representing 63% of the final value, a) was deemed as being sufficiently deviated from the initial linear trend to be defined as the shear stress sensitivity, τ_w^*. This metric was calculated for 7 separate vessel preparations and compared between the 3- and 5-cmH_2O average transmural pressures.

4.4.4 Application of Transaxial Pressure Gradient Sine Functions

In order to study the contractile response of lymphatics to time-varying transaxial pressure gradients, the author used the ELPS to apply the following function:

$$\Delta P = A\sin(2\pi ft) + C \tag{4.3}$$

where A is the amplitude [cmH_2O], f is the frequency [Hz], and C is the offset [cmH_2O]. This waveform was applied after 5 min of no applied pressure gradient with an average transmural pressure of 3 cmH_2O. Two separate conditions were tested on the 5 vessels following the shear sensitivity ramps: $f = 0.0625$ and 0.125 Hz (or 3.75 and 7.5 min^{-1}, which are within the observed range of lymphatics [11, 30]) for 5 min each, both with $A = 0.5$ cmH_2O, $C = 1$ cmH_2O, and a constant average transmural pressure, P_{avg} = 3 cmH_2O. For each condition, the mean flow rate through the vessel was calculated, as well as wall shear stress value over that time. Additionally, the vessel tone was calculated for each sine wave condition, as well as the mean contraction frequency of the sinusoidal condition normalized to the mean contraction frequency of the no-flow control condition (zero axial pressure gradient with an average transmural pressure of 3 cmH_2O for 5 min). For comparison purposes, both the mean shear stress and normalized contraction frequency were calculated for the steady flow control ($\Delta P = 1$ cmH_2O and $P_{avg} = 3$ cmH_2O for 5 min), and the tone was also calculated for the no-flow control condition.

On four separate vessel preparations, the author investigated the vessel contractile response to a sine function of large amplitude. Specifically, a condition consisting of $A = 2$ cmH_2O, $f = 0.125$ Hz (7.5 min^{-1}), and $C = 2$ cmH_2O [Eq. (4.3)] with a constant average transmural pressure of 3 cmH_2O was applied for 3 min. In addition, on two of these vessels, this condition was followed immediately by a similar sine function except with $f = 0.0625$ Hz (3.75 min^{-1}). In these two vessels, the sine waves at these two frequencies were also applied after each vessel was equilibrated in a bath treated with L-NAME (10^{-4} M) for 15 min to check for a nitric oxide (NO) dependency. As in the previously described

conditions prescribed on the vessel, the large-amplitude sine function was preceded by a 5-min condition with no applied pressure gradient and an average transmural pressure of 3 cmH_2O. To analyze the vessel's response to this time-varying axial pressure gradient, the author calculated the point-to-point contraction frequency over time. In this way, the contractile activity of each vessel could be investigated upon the onset of a large, periodic axial pressure gradient waveform.

4.4.5 Statistics

The fluid shear stress sensitivity experiments consisted of $n = 7$ vessels and were compared using a two-tailed paired ratio Student's t-test. As for the transaxial pressure gradient sine function experiments, the normalized contraction frequency, mean shear stress, and tone for the $f = 0.0625$- and 0.125-Hz conditions ($n = 5$) were compared with their respective flow control using a paired one-way ANOVA with a Dunnet multiple-comparison correction. For all cases, significance was defined as $p < 0.05$ (*) or $p < 0.01$ (**).

4.5 Endpoint Metrics for Hypotheses

In order to address *Hypothesis 1*, the author utilized the shear sensitivity metric described in Section 4.4.3, τ_w^*, and compared two conditions: a steady transmural pressure of 3 and 5 cmH_2O. Specifically, both the diameter and estimated flow rate data were used during the ΔP ramp described in Section 4.4.3 to calculate τ_w^* and compare at each P_{avg}. A significant difference between the mean shear stress sensitivities at these two transmural pressures would confirm *Hypothesis 1*. To test *Hypothesis 2*, the small amplitude ΔP sine functions described in Section 4.4.4 were applied at a constant $P_{\text{avg}} = 3$ cmH_2O. In particular, contraction frequency (normalized to a no-flow control) was calculated from recorded diameter data for a steady-flow condition ($\Delta P = 1$ cmH_2O) and the two frequency conditions mentioned in Section 4.4.4. Additionally, mean shear stress was calculated from the measured diameter and estimated flow rate over each applied condition. A significant difference in normalized contraction frequency between the steady flow and sine function conditions

would support *Hypothesis 2* if the mean wall shear stresses were not significantly different among all of the conditions. Finally, to test *Hypothesis 3*, the two large ΔP sine functions described in Section 4.4.4 were applied at a constant $P_{\text{avg}} = 3$ cmH_2O. Using the measured vessel diameter, the contraction frequency was calculated as a function of time (counting primary points of systole) and analyzed before and after the onset of the ΔP sine function. Ultimately, *Hypothesis 3* will be confirmed if both (1) the contraction frequency over time approaches or becomes the applied sine frequency; and (2) the same vessel, when in a passive (calcium-free) state, shows no variations due to induced pressure artifacts from the ELPS when the same condition is repeated.

4.6 Results

Figure 4.1 is an example of the ELPS applying a transaxial pressure gradient ramp from 0 to 3 cmH_2O over 3 min while simultaneously holding the average transmural pressure constant at 3 cmH_2O. The rate of the solenoid valve switching (1 for ON and 0 for OFF), which is indicative of the volume of fluid being moved through the vessel, is also shown. As expected, the pressure gradient ramp also produced a linearly increasing flow rate, which was estimated every 5 sec using the solenoid valve state and the position of the syringes [Section 3.3.3]. Despite the ramped flow rate, the estimated wall shear stress did not increase linearly—it instead increased then began to level off due to the active diameter changes of the vessel (*i.e.* combination of phasic and tonic activity). This phenomenon occurred in every vessel preparation, in addition to the eventual complete inhibition of vessel contractility. To verify the instantaneous Poiseuille flow assumption that the author used, both the Reynolds number and Womersley number were calculated for all cases to be less than 20 and 0.8, respectively, indicating that this shear stress constitutive relationship is perfectly valid since the flow is laminar and may be treated as steady.

Looking at just the estimated shear stress values in Fig. 4.1 both after positive nonzero flow and before complete inhibition ($\sim$30-150 sec), the author performed a least-squares

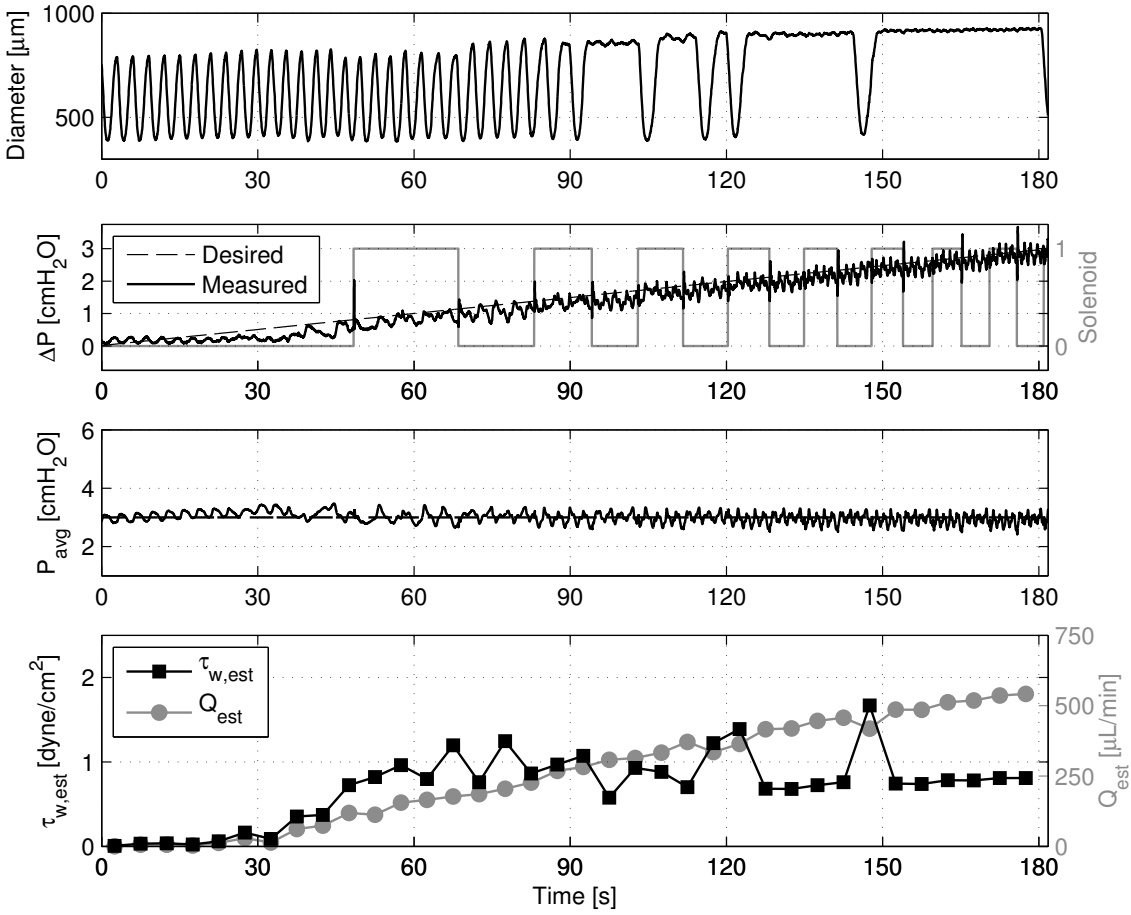


Figure 4.1: Example showing an isolated vessels diameter response to a ΔP ramp going from 0 to 3 cmH_2O over 3 min with a constant $P_{avg} = 3$ cmH_2O. Also shown is the resulting flow rate and shear stress values estimated over sequential 5-sec time windows, as well as the rate of solenoid valve switching.

curve fit to the exponential function in Eq. (4.2), as seen by the representative example in Fig. 4.2. As a reference, a dashed line indicating the initial slope ($1/T_c$ dyne/cm^2-s) of the fitted function is also shown in Fig. 4.2, which roughly indicates the amount of deviation of the shear stress profile from its initial linear trend. The shear sensitivity value, τ_w^* , was found by calculating the shear stress value of the fitted function T_c sec after the start of the data set (which ends up being 63% of the steady-state value of the function, a) [Eq. (4.2)], and this point is denoted by an asterisk in Fig. 4.2. This procedure was applied to each 3-min ramp experiment for all 7 vessel preparations at both $P_{\text{avg}} = 3$ and 5 cmH$_2$O [Table 1]. In all, the mean shear sensitivity of the vessels at an average transmural pressure of 5 cmH$_2$O (1.23 dyne/cm^2) was significantly less ($p < 0.01$) than the shear sensitivity at an average transmural pressure of 3 cmH$_2$O (1.73 dyne/cm^2) [Fig. 4.3].

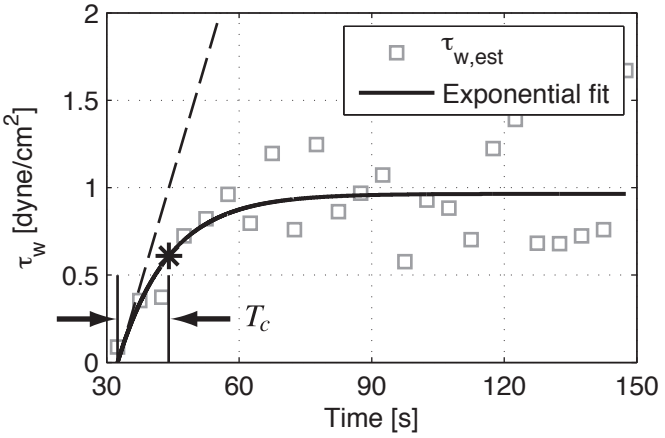


Figure 4.2: Example of the curve-fit using the estimated shear stress values, $\tau_{w,\text{est}}$, from Fig. 4.1 to determine the shear sensitivity metric, τ_w^*, using the parameter, T_c, from the exponential function in Eq. (4.2). The initial slope of the function in Eq. (4.2) (dashed line), $1/T_c$ dyne/cm^2-s, is also shown as a reference, roughly indicating the deviation of the shear profile from its initial linear trend.

With respect to the transaxial pressure gradient sine function experiments, an example of the applied transaxial pressure gradient and resulting vessel diameter may be seen in Fig. 4.4a. In this case, the sine frequency, f, is 0.0625 Hz (3.75 min^{-1}), and the average transmural pressure is held constant at 3 cmH$_2$O for 5 min. This applied sine wave resulted

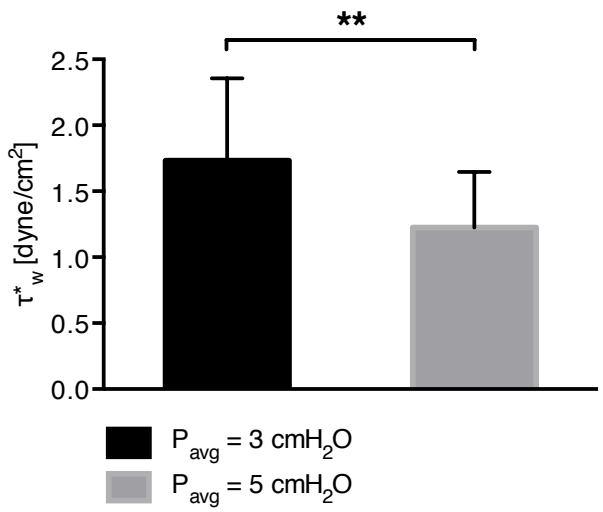


Figure 4.3: The mean shear sensitivity value, τ_w^*, at an average transmural pressure of 5 cmH_2O, 1.23 dyne/cm^2, was significantly less than at an average transmural pressure of 3 cmH_2O, 1.73 dyne/cm^2 ($p < 0.01$, $n = 7$). Error bars represent SEM.

Table 4.1: Shear stress sensitivity values, τ_w^*, calculated for the 7 vessels used in the study at the average transmural pressures of 3 and 5 cmH_2O. As a reference, the calcium-free diameter at 3 cmH_2O is also provided.

		Shear sensitivity [dyne/cm^2]	
Vessel	Diameter [μm]	$P_{\text{avg}} = 3$	$P_{\text{avg}} = 5$
1	655.8	0.47	0.25
2	1062.9	0.83	0.52
3	952.8	0.61	0.62
4	837.9	0.79	0.46
5	606.2	3.66	2.60
6	662.7	1.24	1.14
7	631.1	4.53	2.99

in contraction patterns with a relatively consistent contraction frequency but with time-varying tone and contraction amplitude. Using the solenoid valve switching data, syringe position data, and the vessel diameter data; both the normalized contraction frequency of the vessel and the mean wall shear stress imposed on the vessel was calculated for the steady flow control ($\Delta P = 1$ cmH_2O) and the two sine function conditions ($f = 0.0625$ Hz and $f = 0.125$ Hz, both with $A = 0.5$ dyne/cm^2 and $C = 1$ dyne/cm^2 [Eq. (4.3)]). For all 5 vessels, the mean normalized contraction frequency of the steady pressure gradient control (81.8%) was significantly less ($p < 0.05$) than both the 0.0625-Hz condition (118.0%) and the 0.125-Hz condition (125.6%) [Fig. 4.4b]. However, the mean wall shear stresses across all 5 vessels for the steady ΔP (1.41 dyne/cm^2), 0.0625-Hz (1.12 dyne/cm^2), and 0.125-Hz (0.94 dyne/cm^2) conditions were relatively similar and not significantly different [Fig. 4.4c]. The vessel tone did not exhibit a clear trend in comparison to the frequency data with the 0.0625-Hz (10.2%) and 0.125-Hz (12.2%) conditions being only slightly higher than the steady flow condition (9.53%) and slightly less than the no-flow control (13.0%) [Fig. 4.4d]. None of these conditions were significantly different than the no-flow control ($n = 5$).

Figure 4.5 shows an example of the experiment testing shear coordination where a larger transaxial pressure gradient sine function was applied in conjunction with a constant average transmural pressure of 3 cmH_2O for 5 min. In this instance, the transition is shown between an applied transaxial pressure gradient of $\Delta P = 0$ cmH_2O to the applied sine function where $A = 2$ dyne/cm^2, $f = 0.125$ Hz (7.5 min^{-1}), and $C = 2$ dyne/cm^2 [Eq. (4.3)]. Once ΔP transitions from zero to the prescribed sine function, the vessel diameter also transitions from one relatively consistent tone and contraction frequency to another relatively consistent tone and contraction frequency. Specifically, these vessel contractions begin to rise and fall with the applied transaxial pressure gradient—ultimately matching the frequency of the applied sine function (0.125 Hz). This phenomenon is more clearly seen in Fig. 4.6, which shows the point-to-point contraction frequency over time for four

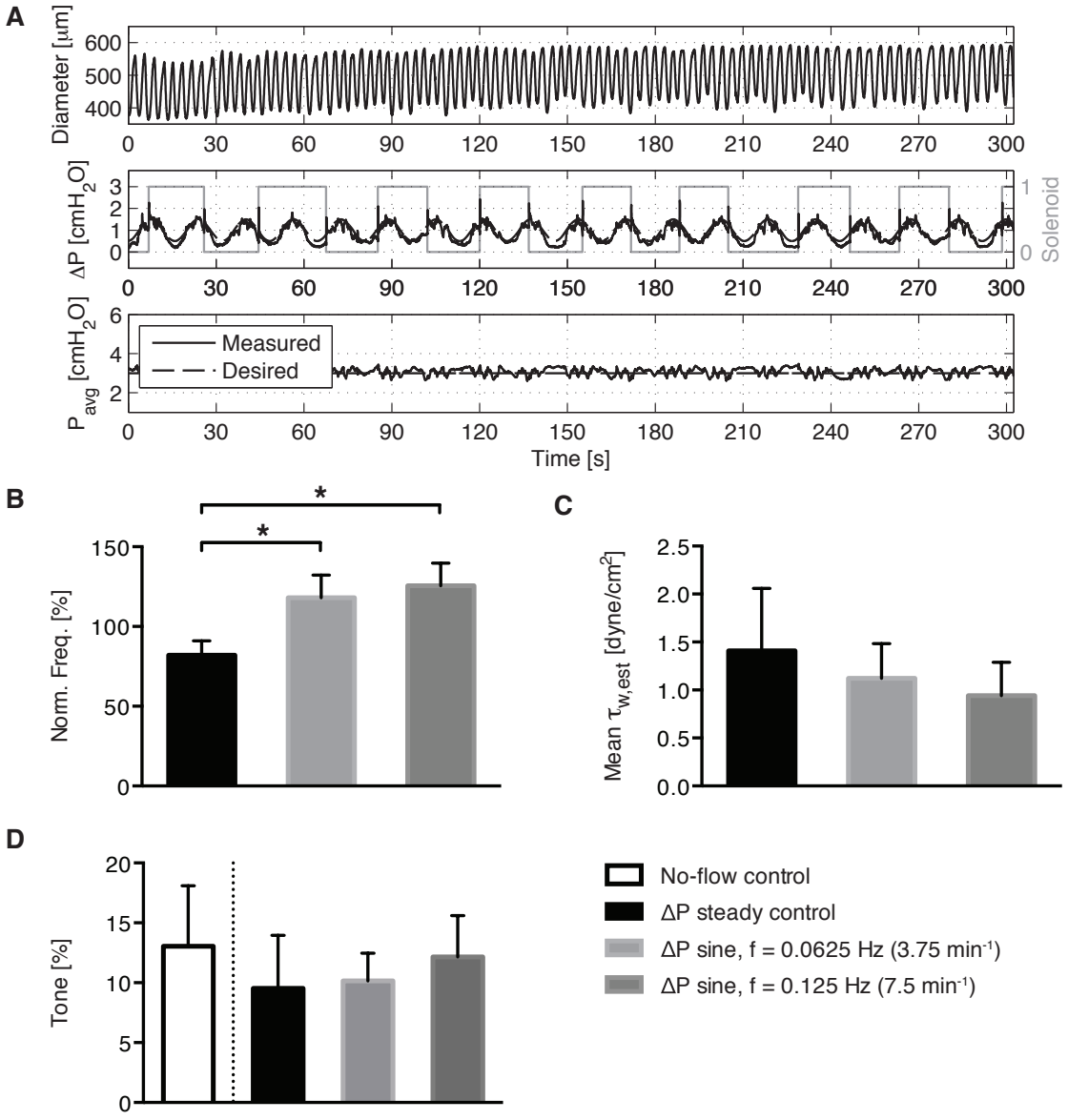


Figure 4.4: (a) Example showing an isolated vessel's diameter response to a ΔP sine function with $f = 0.0625$ Hz (3.75 min^{-1}), $A = 0.5$ dyne/cm^2, and $C = 1$ dyne/cm^2 [Eq. (4.3)] and a constant average transmural pressure of 3 cmH_2O. (b) The mean normalized contraction frequency of the 0.0625-Hz condition, 118.0%, and the 0.125-Hz condition, 125.6%, were significantly greater than the steady ΔP control, 81.8% ($p < 0.05$). Despite this lack of frequency inhibition, (c) the mean wall shear stress across all vessels for the steady ΔP (1.41 dyne/cm^2), 0.0625-Hz (1.12 dyne/cm^2), and 0.125-Hz (0.94 dyne/cm^2) conditions were relatively similar and not significantly different ($n = 5$). (d) The vessel tone for the 0.0625-Hz (10.2%) and 0.125-Hz (12.2%) conditions were only slightly higher than the steady flow condition (9.53%), and the none of these conditions were significantly different the no-flow control (13.0%) ($n = 5$). Error bars represent SEM.

separate vessel preparations before and after the application of the large ΔP sinusoid (resulting in mean shear stresses of 4.03, 2.60, 2.02, and 4.12 dyne/cm^2, respectively, over the condition). In all of these cases, the contraction frequency of each vessel changes relatively quickly from its corresponding resting value to that of the applied sine function.

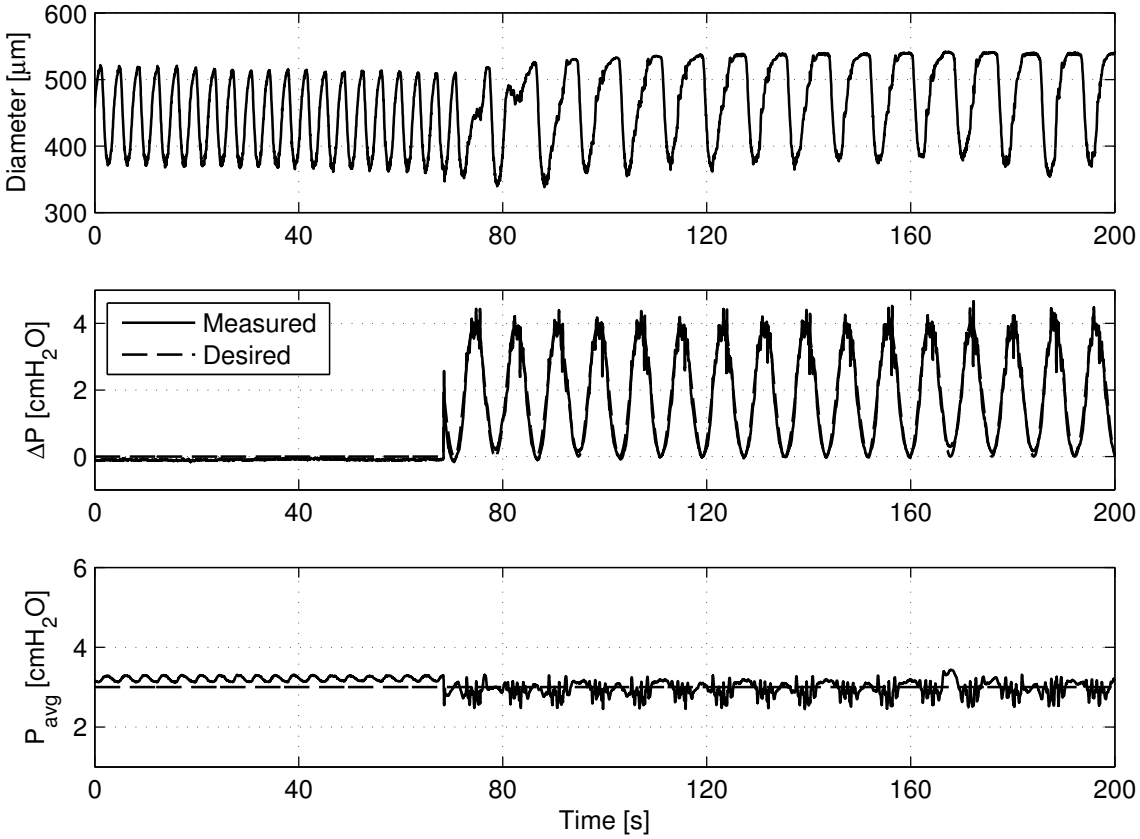


Figure 4.5: Example demonstrating an isolated vessel's diameter response to the onset of a ΔP sine function of large amplitude with $f = 0.125$ Hz (7.5 min^{-1}), $A = 2$ dyne/cm^2, and $C = 2$ dyne/cm^2 [Eq. (4.3)]. After the sine function is applied, the vessel begins to contract at the same frequency.

In addition to the large ΔP sine function at 0.125 Hz, Fig. 4.7 shows the application of a similar sine function but at 0.0625 Hz (3.75 min^{-1}). In this case, the vessel similarly has significant contractions at the same frequency of the applied sign wave. However, due to the slower nature of the sign wave with respect to the $f = 0.125$ Hz case, the vessel is able to attempt several phasic contractions between each significant contraction/inhibition cycle.

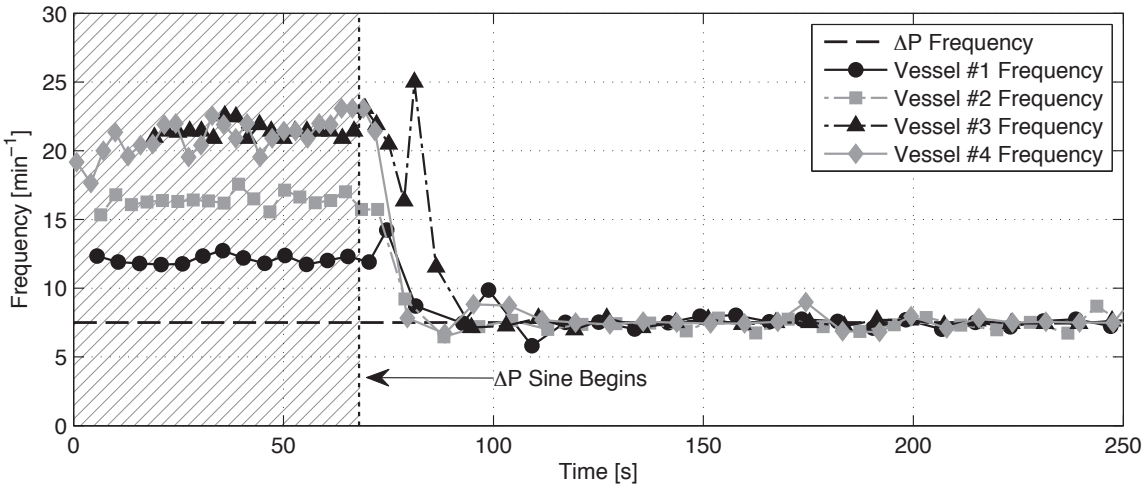


Figure 4.6: Point-to-point frequency data over time for four separate isolated vessel preparations upon the onset of the ΔP sine function from Fig. 4.5. Even though each vessel's contraction frequency differs with no imposed transaxial pressure gradient, they both begin to contract at the same frequency of the sine function once it is applied (7.5 min^{-1}).

Counting just these significant contractions, the contraction frequency over time matches the prescribed ΔP frequency exactly for two separate vessel preparations. To ensure that the vessel responses to these large sine functions were not artificially induced by the ELPS, the $f = 0.125$ Hz case was applied to a vessel equilibrated in calcium-free PSS to eliminate active tone and contractions [Fig. 4.8]. Though the applied ΔP sine function resulted in contractions of similar frequency [Fig. 4.8a], after the application of calcium-free PSS the passive vessel diameter was not affected by the condition [Fig. 4.8b]. Hence, the vessel syncing observed was confirmed to be an active response from the vessel.

4.7 Discussion

As mentioned previously, one of the primary reasons investigators have been unable to answer questions regarding shear stress sensitivity and shear stress dynamics in collecting lymphatic vessels is the lack of adequate experimental tools. In this respect, the ELPS has the unique ability to independently control the two primary mechanical stimuli imposed on a lymphatic vessel: average transmural pressure, P_{avg}, which affects circumferential

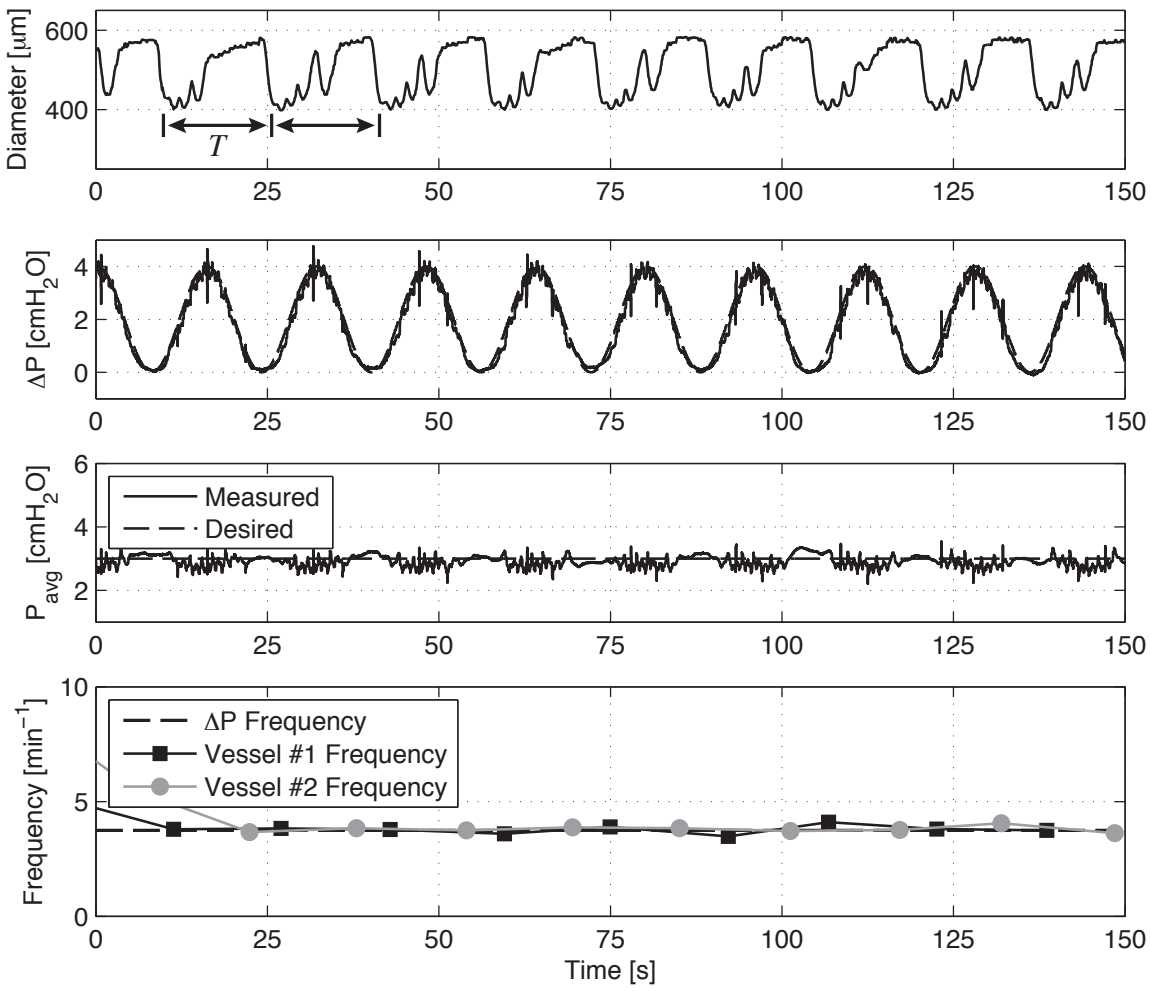


Figure 4.7: Example demonstrating an isolated vessel's diameter response to a sine function of large amplitude with $f = 0.0625$ Hz (3.75 min^{-1}), $A = 2$ dyne/cm^2, and $C = 2$ dyne/cm^2 [Eq. (4.3)]. Similar to before, the vessel has significant contractions at the same frequency of the applied sine wave; however, due to its slower nature the vessel is able to attempt several phasic contractions between each significant contraction/inhibition cycle. Counting just these significant contractions, the contraction frequency over time exactly matches the prescribed ΔP frequency (3.75 min^{-1}) for two separate vessel preparations.

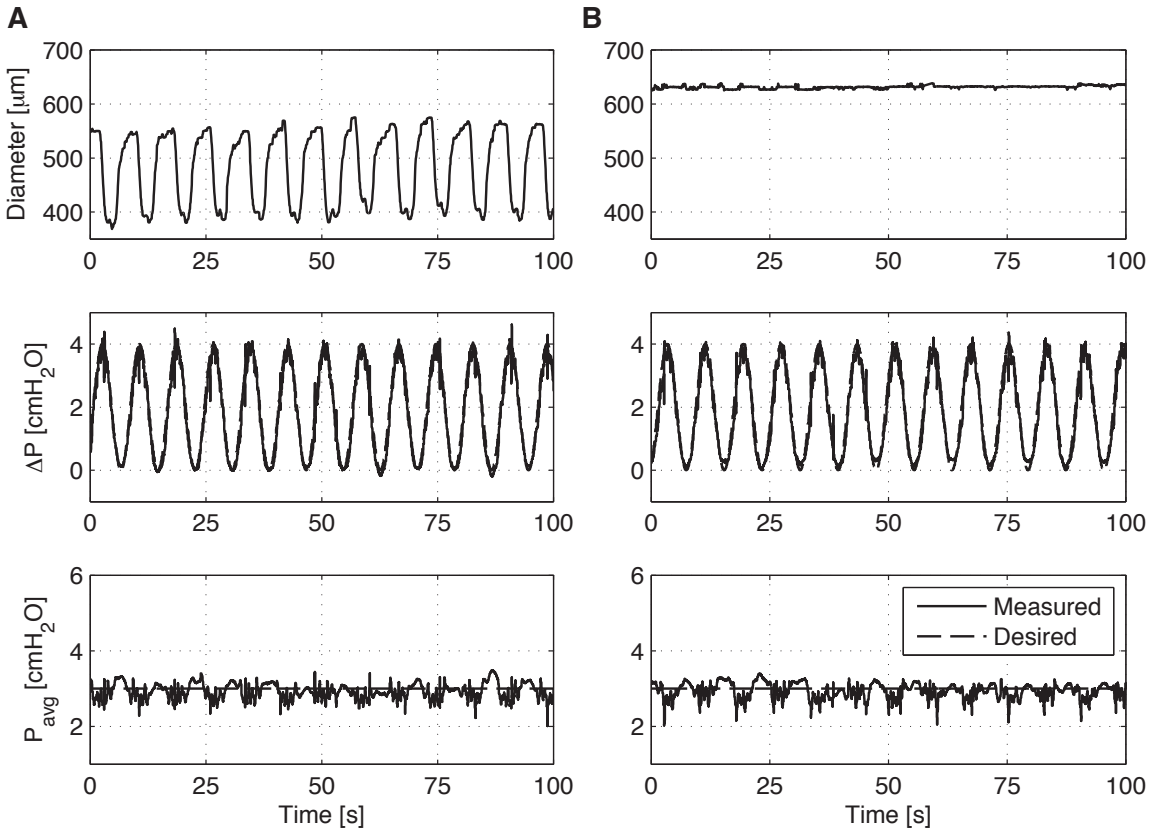


Figure 4.8: To ensure that the vessel syncing was not artificially induced by the ELPS, a sine function of large amplitude with $f = 0.125$ Hz (7.5 min^{-1}), $A = 2$ dyne/cm^2, and $C = 2$ dyne/cm^2 [Eq. (4.3)] was applied to a vessel (a) with PSS and then (b) with Ca^{2+}-free PSS. (a) Though the applied ΔP sine function resulted in contractions of similar frequency; (b) after the application of Ca^{2+}-free PSS, the passive vessel diameter was not affected by the condition.

stress; and transaxial pressure gradient, ΔP, which affects fluid shear stress via flow rate [Chapter 3]. This capability is significant because independent tracking control affords the ultimate flexibility to perform a wide range of experiments, including single-factor studies that aim to isolate shear-mediated mechanisms. Additionally, the ability to measure flow rate through the vessel allows investigators to calculate wall shear stress values empirically for the first time. The author utilized these capabilities in this study to investigate several unanswered questions, including how the shear stress sensitivity in these vessels responds to differing transmural pressures, whether a continuously-varying shear stress waveform can impede frequency inhibition controlling for transmural pressure, and whether shear stress itself can dynamically coordinate contraction activity.

In order to quantify approximate levels of shear stress sensitivity, the author first needed to propose an objective metric to determine the magnitude of shear stress at which contraction inhibition first occurs. In brief, the author utilized the linear relationship between wall shear stress and flow rate in the Poiseuille flow equation [Eq. (4.1)] to define the metric: because a linearly-increasing flow rate (produced by an applied linearly-increasing ΔP) would also result in a linearly-increasing shear stress given the contractile activity (average diameter) remains constant, any deviations in average diameter from this linear trend would indicate when the vessel is actively adjusting its contractile activity both phasically and tonically. This *modus operandi* is advantageous since it includes effects from contractile amplitude, frequency, and tone—all of which affect average diameter of the vessel. Moreover, during contractile inhibition each of these parameters is decreased, collectively increasing the average the diameter and reducing the estimated wall shear stress. Hence, instead of evaluating each one of these variables over time to determine when the vessel begins to respond, this procedure simplifies the task of determining the vessel's shear sensitivity using only one metric.

Once the transaxial pressure gradient ramps were applied, the vessels responded similarly in every case. Specifically, as the flow rate (hence shear stress) increased, the inhibitory effects of shear would cause the average diameter to increase, thus resulting in a leveled-off estimated shear stress [Fig. 4.1]. Thus, the author found that this trend matched Eq. (4.2) quite well [Fig. 4.2] and allowed for an objective shear sensitivity metric to be defined and subsequently compared between different P_{avg} conditions for each vessel. Though these levels of shear sensitivity have remained unknown in the isolated vessel literature, the shear sensitivities found [Table 1] do corroborate with shear stress ranges found in previous studies. In particular, Kawai *et al.* found that eNOS activation in LECs (via calcium flux from the P2X/2Y receptor) occurred above the threshold of 0.5 dyne/cm^2 [31]. Furthermore, Dixon *et al.* measured the average fluid shear stress *in vivo* (0.64 dyne/cm^2) within the rat mesentery [11], and Rahbar *et al.* recently reported an average shear stress of 0.12 and 1.5 dyne/cm^2 for both a control and edemagenic stress condition, respectively, within the rat mesentery *in vivo* [51]. Additionally, given that the vessel cannulation chamber for this set-up is similar to the set-up used in the previously reported flow inhibition studies [19, 18], the applied transaxial pressure gradients utilized to inhibit flow were likely well within the physiologic range of wall shear stress experienced by lymphatics *in vivo*.

Overall, the shear sensitivity at an average transmural pressure of 5 cmH_2O was found to be significantly less than at 3 cmH_2O [Fig. 4.3], confirming *Hypothesis 1*. In addition to confirming interaction between effects of shear stress and transmural pressure (as in previous isolated blood vessel studies [34, 60]), this result could be important to normal contractile function in collecting lymphatics. Specifically, upon the onset of increased intraluminal pressure from lymph formation, this passive diameter would subsequently increase, thus inherently decreasing the average shear stress experienced by the vessel for the same amount of fluid flow [Eq. (4.1)]. Accordingly, assuming that it would be functionally advantageous for the vessel to sense the amount of fluid flow, lowering the shear sensitivity of the vessel during an increased transmural pressure could allow the vessel to maintain

a similar sensitivity to flow rate even though the average diameter has increased. Further investigation is needed to provide better insight into the physiological mechanisms of this phenomenon.

This difference in shear stress sensitivity at different transmural pressures is particularly interesting in the context of lymphatic pathologies that alter the mechanical environment surrounding the vessels. One such pathology is lymphedema, a disease characterized by gross swelling of the tissue estimated to affect over 130 million people worldwide [54, 55]. Lymphedema has been shown to result in elevated transmural pressure in the effected area [23], and the accumulation of fibrotic tissue and adipocytes due to lymph stasis can drastically alter the mechanical environment surrounding the vessels [56]. Taken together, these results suggest that chronic disruption of the mechanical state of the collecting lymphatic vessels could have a significant effect on their shear stress sensitivity. Whether a loss of shear sensitivity is detrimental to lymphatic function remains experimentally unexplored; however, computational models have suggested that it is important in situations of elevated lymph flow [64].

Furthermore, investigators have recently shown that the unique biomechanical environment of lymphatics is essential in guiding its development [5, 57]; thus, the presence of chronically high pressures could result in shifts of shear sensitivity toward another phenotype, such as veins. However, extrapolating these results to pathological cases remains difficult since current data on the mechanical sensitivity of lymphatics has only been collected across relatively short time scales. Like mechanically-induced growth and remodeling displayed in the blood vasculature over days and weeks [28], lymphatic vessels have also been shown to remodel in response to pathological levels of mechanical loading [13, 42, 5]. Hence, future work in this area should focus on longer-term studies relating the consequences of vessel remodeling in lymphatic disease to the mechanosensitivity of the intrinsic lymphatic pump.

With respect to dynamically applied shear, the transaxial pressure gradient sine functions did not result in frequency inhibition for both the 0.0625-Hz and 0.125-Hz (3.75-min^{-1} and 7.5- min^{-1}) conditions with respect to the steady flow control ($\Delta P = 1$ cmH_2O) [Fig. 4.4b]. This result is interesting in light of Fig. 4.4c: even though the mean imposed shear stresses for the 0.0625-Hz and 0.125-Hz conditions were not significantly different than the steady flow control, only the control condition of steady imposed flow resulted in frequency inhibition. Thus, the presence of a continuously varying transaxial pressure gradient within a physiologic frequency range (< 1 Hz [11]) seems to impede the typical frequency inhibition that occurs in response to a steady shear stress, ultimately confirming *Hypothesis 2*. This result would also support the recent *in vivo* findings of Rahbar *et al.*, who found that, in a rat model of edemagenic stress, the contraction frequency did not decrease when compared to control—though the wall shear stress was almost an order of magnitude higher [51]. The results in this current study, which also controls for average transmural pressure, also confirms the original observation by Gashev *et al.* that pressure gradient transitions may temporarily increase contraction frequency in isolated lymphatic vessels [19]. As originally proposed in that paper, this lack of phasic contractile inhibition could be beneficial during normal pump function, where each lymphangion experiences rapidly-changing flow rates and wall shear stresses.

However, unlike what Gashev *et al.* proposed, transaxial pressure gradient (or fluid shear stress) reversal was unnecessary to observe this effect on contraction frequency. Though it has been previously established that collecting lymphatics are exposed to oscillatory flow rates during normal contractile function [11, 30], the two ΔP sine functions that the author imposed [Fig. 4.4] were strictly positive and contained no flow reversal. Hence, reversal of flow direction was unnecessary to impede the phasic frequency inhibition normally observed in response to fluid shear stress. Since Gashev *et al.* only observed this fast chronotropic response when the transaxial pressure was altered after flow was already established (in either direction, orthograde or retrograde), perhaps fluid deceleration could

play a role in these observations. With respect to tone, the author found that the sine function conditions had only slightly increased tone compared to the steady-flow control and had little difference in value compared to the no-flow control [Fig. 4.4d], indicating that the effects on tone are not as substantial as the affects on phasic contraction frequency. Further studies are necessary (both *ex vivo* and in vitro) in order to elucidate these mechanisms.

In addition to the ΔP sine functions applied in Fig. 4.4 ($A = 0.5$ dyne/cm^2 and $C = 1$ dyne/cm^2 [Eq. (4.3)]), the author applied a sine wave of larger amplitude ($A = 2$ dyne/cm^2, $f = 0.125$ Hz, and $C = 2$ dyne/cm^2 [Eq. (4.3)]) on four separate vessel preparations to test the potential presence of contractile coordination via dynamic shear. Accordingly, the author observed contraction frequency patterns transition from one value (unique to that vessel at no applied transaxial pressure gradient) to the exact frequency of the applied ΔP sine function upon the onset of the condition [Fig. 4.5, Fig. 4.6], confirming the first portion of *Hypothesis 3*. Although the four vessels studied did not exhibit similar contraction amplitude trends, contractile tone was decreased in all cases and contractile events were synchronized with the applied pressure gradient waveform while controlling for the average transmural pressure. In addition, this syncing effect was not just unique to this ΔP frequency as it was seen in the two vessels that were also exposed to a ΔP sine function at 0.0625 Hz (3.75 min^{-1}) [Fig. 4.7]. Moreover, this phenomenon is not a passive artifact of the ELPS operation since the same waveform did not cause diameter changes in a vessel after the application of calcium-free media [Fig. 4.8], confirming the second part of *Hypothesis 3*. Hence, for the first time the author has demonstrated that dynamic shear stresses alone have the ability to independently coordinate contractile activity.

This capability could be very significant given the widely varying waveform patterns observed in the lymphatics *in vivo* [11]: considering the previously-shown interaction between shear sensitivity and transmural pressure in this current study, dynamic shear stresses exerted on the endothelium could also play a complementary (or possibly essential) role in

coordinating lymphatic pump function to optimize lymph transport. Accordingly, previous evidence supports this idea of dynamic fluid shear stress playing a significant role in lymphatic contractile coordination. Specifically, Bertram *et al.* have demonstrated computationally that the phase difference between individual contractions along a chain of lymphangions is a significant parameter in the overall pumping efficiency [2], suggesting that some level of coordination among vessel segments would be very beneficial. Consequently, McHale and Meharg showed that bovine lymphatic vessel segments are capable of contractile entrainment and hypothesized that the electrical coupling between lymphangions could play an important role [41]. Moreover, it has been shown that 80-90% of rat lymphatic contractions are coordinated between adjacent lymphangions, and this coordination is mostly the result of endothelial gap junction communication [68]. However, despite this information, additional details of the biophysical and biological mechanisms responsible for this coordination have eluded researchers. The results in this current study provide direct evidence that fluid shear stress has the capacity to regulate the coordination of lymphatic pumping activity between lymphangions, further supporting a previous hypothesis wall shear stress is an essential self-regulatory mechanism in the rat thoracic duct for optimizing lymphatic pumping [20]. However, in contrast to this study that showed NO as a primary regulatory element for tone, initial experiments that block nitric oxide sythanse through the application of L-NAME did not prevent the frequency-following effect [Appendix C.1; Fig. C.1]. Hence, further study is needed in order to investigate the mechanism behind these observations.

In short, these new capabilities provided by the ELPS to both estimate fluid shear stress and independently (and dynamically) vary ΔP and P_{avg}—the factors modulating the two primary mechanical forces exerted on lymphatics: fluid shear stress and circumferential stress—allow for new single-factor studies to be performed that would be nearly impossible with *in vivo* models. In particular, the ability to dynamically vary ΔP (hence shear stress) while controlling for average transmural pressure has allowed the author to confirm

the existence of interactions between P_{avg} and shear sensitivity (*Hypothesis 1*), verify the differential effects of dynamically varying shear on contraction frequency (*Hypothesis 2*), as well as discover the existence of shear-specific contractile coordination for the first time (*Hypothesis 3*). In addition to providing these interesting discoveries and insights into the functional role of shear stress on the collecting lymphatics, the author believes the ELPS provides a promising platform to perform new studies aimed at isolating the affects of fluid shear stress and transmural pressure. By unraveling these difficult and complex interactions comprising the biomechanical state of these vessels, the author hopes this platform will spark novel discoveries in the future with regards to both physiologic and diseased conditions of lymphatic fluid transport.

CHAPTER V

CONCLUSION

This thesis presents both tools and methodologies that have been used to characterize lymphatic pump function in response to dynamic mechanical loading. On the *in vitro* side, the author has developed a low-cost, reliable microcontroller platform capable of augmenting the functionality of conventional peristaltic pumps to apply arbitrary flow waveforms. Specifically, the Arduino Uno, a microcontroller development board, is used to provide open-loop control of a digital peristaltic pump using precisely-timed serial commands. In addition, the flexibility of this platform is further demonstrated through its support of a custom-built cell-straining device capable of producing oscillatory strains with varying amplitudes and frequencies. In this way, the door is opened for future studies whose purpose is to study the cellular mechanisms of LECs exposed to the highly dynamic and complex waveforms typically experienced *in vivo*.

Additionally, the author has designed, constructed, and tested an *ex vivo* lymphatic perfusion system (ELPS) that can independently control both transaxial pressure gradient and transmural pressure—the factors modulating the two primary mechanical forces exerted on lymphatics: fluid shear stress and circumferential stress—dynamically within the lumen of an isolated lymphatic vessel. In this way, the user can modulate the biomechanical state of the vessel, allowing for true single-factor studies to be performed that would be nearly impossible with *in vivo* models. To achieve this control on the vessel, the described system independently actuates two glass syringes on either end of the vessel with precise linear stages. Furthermore, a custom embedded control platform was designed and built (based on the Arduino Uno platform mentioned previously) to support the system using a linear, explicit model predictive control (MPC) algorithm.

In brief, the ability to dynamically vary ΔP (hence shear stress) while controlling for average transmural pressure has allowed the author to explore areas of lymphatic biomechanics that researchers have remained unable to investigate, including the interactions between P_{avg} and shear sensitivity as well as the effects of dynamically varying shear on contraction frequency and coordination. In doing so, the author has made several key observations: transmural pressure affects the shear sensitivity of the lymphatic endothelium, with a higher transmural pressure increasing shear stress sensitivity; and time-varying shear stress has the ability to obstruct inhibition of phasic contraction frequency and even coordinate contractions—providing the first evidence that dynamic shear could play a such an important role in the normal contractile function of collecting lymphatic vessels. In addition to providing these groundbreaking insights into the functional role of shear stress in the collecting lymphatics, the author believes the ELPS provides a promising platform to perform new studies aimed at isolating the affects of fluid shear stress and transmural pressure. By unraveling these difficult and complex interactions comprising the biomechanical state of these vessels, the author hopes this platform will spark more novel discoveries in the future with regards to both physiologic and diseased conditions of lymphatic fluid transport.

5.1 Contributions

In summary, the specific contributions of this thesis are outlined below:

- Developed a multi-purpose, reliable and low-cost electronics platform for studying lymphatic biomechanics *in vitro* using the Arduino Uno microcontroller (Chapter 2).
- Designed, built and tested an *ex vivo* lymphatic perfusion system (ELPS) capable of independently controlling the two primary mechanical stimuli imposed on a lymphatic vessel: average transmural pressure, P_{avg}, which affects circumferential stress; and transaxial pressure gradient, ΔP, which affects fluid shear stress via flow rate (Chapter 3).

- Implemented a control scheme capable of both achieving independent control of the transaxial pressure gradient and transmural pressure, as well as estimating the fluid shear stress, within an isolated lymphatic vessel using embedded hardware. Specifically, a linear, explicit model predictive control (MPC) algorithm was used and implemented on a chipKIT Uno32 microcontroller development board (Chapter 3).

- Quantified shear stress sensitivity in the collecting lymphatics (specifically, the rat thoracic duct) and determined if and how it is affected by an elevated transmural pressure (Chapter 4).

- Confirmed the idea that continuously-varying pressure gradient (*i.e.* fluid shear stress) waveforms within a physiologic frequency range can obstruct the typical flow-induced inhibitory effect on contraction frequency—independent of average transmural pressure (Chapter 4).

- Discovered that a dynamic fluid shear stress alone has the capability to coordinate lymphatic contractile activity while controlling for average transmural pressure (Chapter 4).

5.2 Future Work

Given the current work summarized in this thesis, the author believes that future research directions may be divided into two main categories: experimental and computational. With respect to experimental work (as reiterated previously), the main goal is to understand how dynamic forces affect lymphatic pump function, which on an individual (lymphangion) level could then be potentially extrapolated into how the general network responds to diverse, locally-applied loads. Although this thesis provides additional insight, the author's results leave several remaining questions: for instance, what is the cellular mechanism by which transmural pressure alters shear stress sensitivity, and could this same mechanism be externally altered to enhance sensitivity in regions where it might have been degraded

(*e.g.* in pathologies like lymphedema)? Also, what is the mechanism that keeps dynamic shear stress from inhibiting the intrinsic lymphatic pump as in steady-flow scenarios, and to what extent does shear stress play a role in contractile coordination and regulation among lymphangions? Consequently, could this mechanism also be externally adjusted to stimulate pump function (and potentially enhance lymph transport) in lymphatic diseases such as lymphedema?

As for computational approaches, the author believes that optimal control approaches (like in Section 3.3.2) may be used to gain better insight into the contractile responses of an individual lymphangion to dynamic mechanical loads. Specifically, by taking a computational model of a lymphangion (the "plant"), one could solve for (in reverse) the optimal contraction patterns that minimize a particular cost function (*i.e.* control objective) and determine whether this behavior fits experimental observations. In this way, it would be possible to ascertain whether or not the lymphangion aims to satisfy a certain objective. For instance, if one hypothesized that the intrinsic lymphatic pump behaves as to maximize lymph transport while minimizing expended energy (and thus described a corresponding cost function mathematically), previously-performed experimental scenarios could be recreated computationally to see if the virtual vessel responds in a similar way. Likewise, other cost functions could be tested, such as whether or not the vessel seeks to maintain a particular biomechanical state or tries to minimize effort with respect to a particular portion of the network. Not only would this capability allow for more realistic computational lymphatic models, but this higher-level understand could enhance researchers' understanding of the mechanisms governing this activity. Moreover, much more complex relationships among lymphangions in a network could be explored computationally, allowing researchers the opportunity to perform virtual experiments that would be almost impossible to conduct with *in vitro*, *in vivo*, or *ex vivo* models.

APPENDIX A

ANCILLARY MATERIAL FOR CHAPTER II

A.1 Sending Serial Commands to Peristaltic Pump

```
# pump_control.py

# Load necessary modules
import time
import serial
import struct
from numpy import *

# Define various variables and parameters; (**)-modify these
CR = chr(13)                        # carriage return
exptime = 24                        # experiment time, hr **
freq = 0.1                          # frequency, Hz **
w = 2*pi*freq                       # set omega
period = (1/freq)*1e3               # set period (ms)
cycles = ceil(3600*exptime*freq)# number of period cycles
deltat = 90.                        # set delta t, ms **
t = arange(0.,period/1e3,deltat/1e3)   # set time steps

# Set flow parameters and function
l = 4.5                             # chamber width, mm
a = 0.4                             # chamber height, mm
mu = 6.92e-4                        # water viscosity @ 37C, Pa-s

tau = 2*sin(w*t) + 1                # desired shear stress function,
    dyne/cm^2
f = (tau*l*a**2)/(1e3*mu)           # corresponding flow rate, mL/min
f = 12.01*f                         # corresponding RPM (for 1.3mm ID
    tubing)


# ----- Begin main function ----- #
def main():
    # Load up command array with initial info:
    #   period (word), cycles (word), deltat (word)
    #  First, break the words into bytes
    period8 = struct.pack('>L', period)[2:]
    cycles8 = struct.pack('>L', cycles)[2:]
    deltat8 = struct.pack('>L', deltat)[2:]

    commands = [ ord(period8[0]), ord(period8[1]),
            ord(cycles8[0]), ord(cycles8[1]),
            ord(deltat8[0]), ord(deltat8[1]) ]
```

```
    # Load up speed commands
    speeds = loadSpeeds(f)

    # Load entire command array, including unique command initiation
        sequence
    commands = [ ord('$'), ord('$') ] + commands + speeds

    # Open serial connection
    s = serial.Serial(
            port='/dev/tty.usbmodem411',
            baudrate=9600,
            timeout=30
        )

    # Send data
    print("Sending data...\n\n")
    s.open()
    s.isOpen()
    # ...one "byte" at a time!
    i = 0
    for command in commands:
        # Write a byte
        s.write(chr(command))

        # Pause so microcontroller can keep up
        time.sleep(15e-3)

        # Pause extra time after first byte
        if i == 0:
            time.sleep(2)
            i += 1

    # Done!
    print("Data sent.\n")
    s.close()


# ----- Convert numerical rpm value to serial string ----- #
def loadSpeeds(waveform):
    f = waveform
    length = len(f)      # length of waveform

    # Figure out what to do for first command
    #  Find dir change and slope of first point
    dirchange = sign(f[1]*f[0])
    slope = sign(f[1]-f[0])

    # Set initial speed dir based on if dir change
    if dirchange > 0:
        if f[0] > 0:
            rpm_array = [ord('J')]     # CW dir
        else:
            rpm_array = [ord('K')]     # CCW dir
```

```
elif dirchange < 0:
    if slope > 0:
        rpm_array = [ord('J')]
    else:
        rpm_array = [ord('K')]
else:
    if slope > 0:
        rpm_array = [ord('J')]
    else:
        rpm_array = [ord('K')]

# Figure out rest of signal
for i in xrange(1,length-1):
    # Find dir change and slope
    dirchange = sign(f[i+1]*f[i])
    slope = sign(f[i+1]-f[i])

    # Dir change? Tack on dir change command
    if dirchange < 0:
        if slope > 0:
            rpm_array.append(ord('J'))
        else:
            rpm_array.append(ord('K'))

    if dirchange == 0:
        if slope > 0:
            rpm_array.append(ord('J'))
        else:
            rpm_array.append(ord('K'))

    # Now, convert value to correct speed command
    #  Decompose number into whole # and decimal
    whole = floor(abs(f[i]))
    dec = round((abs(f[i])-whole)*100)

    # Decimal round up?
    if dec == 100:
        whole += 1
        dec = 0

    # Less than 1 rpm? Keep it at 1 rpm!
    if whole == 0:
        whole = 1
        dec = 0

    # Send whole # and decimal part of speed, separately
    rpm_array.append(ord('S'))
    rpm_array.append(uint8(whole))
    rpm_array.append(uint8(dec))

# Return array of rpms
return rpm_array
```

```
# ----- Necessary pythonic stuff ----- #
if __name__ == "__main__":
    main()
```

```
// pump_arduino_code.pde

// Include lcd library, EEPROM library, & stdio
//  => For use with standard 16x2 LCD

#include <LiquidCrystal.h>
#include <EEPROM.h>
#include <stdio.h>

// Connections:
//  rs (LCD pin 4) to Arduino pin 12
//  rw (LCD pin 5) to Arduino pin 11
//  enable (LCD pin 6) to Arduino pin 10
//  LCD pin 15 to Arduino pin 13
//  LCD pins d4, d5, d6, d7 to Arduino pins 5, 4, 3, 6
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);

// Define carraige return ASCII
#define CR 13
#define SerialPower 3

// Define some other variables
boolean flag = false;           // set first command flag
unsigned int countCR = 0;       // set counter variable
unsigned int addr = 0;          // set initial EEPROM address
unsigned int addrMem;           // addr pulled from memory
unsigned int i, j, k;           // for loops
unsigned long timeDatum = 0;    // define timer datum
char blankLine[] = "                ";  // set blank line string

  // Create buffer vars to store printed commands
  char buff1[3];    //  for dir changes
  char buff2[10];   //  for speed changes
byte someByte;      //  for storing byte read from memory

// Start main function to run once before looping
void setup()
{
  // Make sure serial converter power is off during setup
  pinMode(SerialPower, OUTPUT);
  digitalWrite(SerialPower, LOW);

  // Print initial LCD stuff
  lcd.begin(16,2);              // columns, rows.  use 16,2 for a 16
     x2 LCD, etc.
  lcd.clear();                  // start with a blank screen
  lcd.setCursor(0,0);           // set cursor to column 0, row 0 (
     the first row)
  lcd.print("Waiting...");      // change this text to whatever you
     like. keep it clean.
```

```
// Begin serial data
Serial.begin(9600);
delay(100);

// For 20 sec, wait for serial...then start!
timeDatum = millis();  // set time datum
while ( millis()-timeDatum < 30*1000 )
{
  // Listen for some new serial data
  if (Serial.available() > 0)
  {
    // Heard something
    someByte = Serial.read();

    // Should I continue listing?
    if (flag == false && someByte == 36)
    {
      // Set flag active to start listening!
      flag = true;
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Receiving...");

      continue;  // skip to next serial command
    }

    // Listening, so store command data
    if (flag == true)
    {
      // Store data on memory
      //   +2 is to leave memory bank (0, 1 open)
      EEPROM.write(addr+2,someByte);
      addr++;
    }
  }
}

// Memory written? Store how much! (in 0, 1)
//   ...then prompt to turn off
if (addr > 0)
{
  EEPROM.write(0,highByte(addr+2));
  EEPROM.write(1,lowByte(addr+2));
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Data received.");
  lcd.setCursor(0,1);
  lcd.print("*** Turn off ***");
  // stall forever HAHAHA!
  while (1)  delay(1000);
}

// OK, ready? Go!
```

```
digitalWrite(SerialPower, HIGH);

// Display on LCD that it's calibrating
lcd.clear();
lcd.print("Calibrating...");
lcd.setCursor(0,1);

// Confirm input serially

// Pull addr slot from memory
byte addrHB = EEPROM.read(0);
byte addrLB = EEPROM.read(1);
addrMem = byteCombine(addrHB,addrLB);

// Pull all of memory into an array
byte romValues[addrMem+1];
for (i=0; i<addrMem; i++)
  romValues[i] = EEPROM.read(i);

// Pull time delay from memory in ms
unsigned int deltaT = byteCombine(romValues[6],romValues[7]);

/* FOR DEBUGGING STUFF ON EEPROM
###################
for (i=0; i<addrMem; i++)
{
  Serial.print(romValues[i], DEC);

  Serial.print('\n');
  //Serial.println();
  delay(30);
}
###################
*/

// Calibration!
// Print out commands using string buffers, and let's time it

// Put pump into RPM mode
Serial.println("1L");
delay(deltaT);

// Set initial direction and speed
printSpit(8, 12, deltaT, romValues);

// Start pump
Serial.println("1H");
delay(deltaT);

timeDatum = millis();  // set time datum
for (i=0; i<3; i++)
{
  // Print how many cycles left
  lcd.print(blankLine);
```

```
  lcd.setCursor(0,1);
  lcd.print(3-i);

  // Send function point
  printSpit(12, addrMem, deltaT, romValues);
}

// Calculate actual time that process took
unsigned int actualT = millis()-timeDatum;
// Pull desired period from memory
unsigned int desiredT = byteCombine(romValues[2],romValues[3]);

// Stop pump
Serial.println("1I");
delay(deltaT);

// Calculate new deltaT
float x = 3*(float)desiredT/(float)actualT;
unsigned int deltaTnew = (unsigned int)(x*(float)deltaT);

/* DEBUGGING CALIBRATION
################
// print out how it took to print a function period (in ms)
Serial.print("\n\nTotal time: ");
Serial.print(actualT);
Serial.print(" ms\n");
Serial.print("New deltaT: ");
Serial.print(deltaTnew);
Serial.print(" ms\n\n");

// retest with calibrated deltaTnew
timeDatum = millis();
printSpit(7, deltaTnew, romValues);
actualT = millis()-timeDatum;

// print out new junk
Serial.print("\n\nNew total time: ");
Serial.print(actualT);
Serial.print(" ms\n\n");

################
*/

// Now cycle thru all function cycles!
lcd.clear();
lcd.print("Cycles left:");
lcd.setCursor(0,1);
delay(100);

unsigned int cycles = byteCombine(romValues[4],romValues[5]);
unsigned int cycles_datum = cycles;

// Set initial direction and speed
printSpit(8, 12, deltaTnew, romValues);
```

```
  // Start pump!
  Serial.println("1H");
  delay(deltaTnew);

  for (i=0; i<cycles; i++)
  {
    // How many cycles left?
    lcd.print(blankLine);
    lcd.setCursor(0,1);
    lcd.print(cycles_datum-i);

    // Output function point
    printSpit(12, addrMem, deltaTnew, romValues);
  }

  // Stop pump
  Serial.println("1I");

  // Done!
  lcd.clear();
  lcd.print("Done!");
}

// Recursive loop occurring when complete (do nothing)
void loop()
{
  delay(100);
}


// Stick together a highbyte & lowbyte into a word
unsigned int byteCombine(byte HB, byte LB)
{
  return (HB << 8) | LB;
}

// Function to spit out serially
void printSpit(unsigned int startAddr, unsigned int endAddr,
        unsigned int DeltaT, byte *RmValues)
{
  for (k=startAddr; k<endAddr; k++)
  {
    switch (RmValues[k])
    {
      case 74: // J (CW)
        sprintf(buff1, "1%c", RmValues[k]);
        Serial.println(buff1);
        break;
      case 75: // K (CCW)
        sprintf(buff1, "1%c", RmValues[k]);
        Serial.println(buff1);
        break;
      case 83: // S (speed command)
```

```
        sprintf(buff2, "1S%04u%02u", RmValues[k+1], RmValues[k+2]);
        Serial.println(buff2);
        k = k+2;
        break;
      default:
        continue;
    }
    // Pause for correct amount of time (ms)
    delay(DeltaT);
  }
}
```

A.2 Cell Straining Device Control

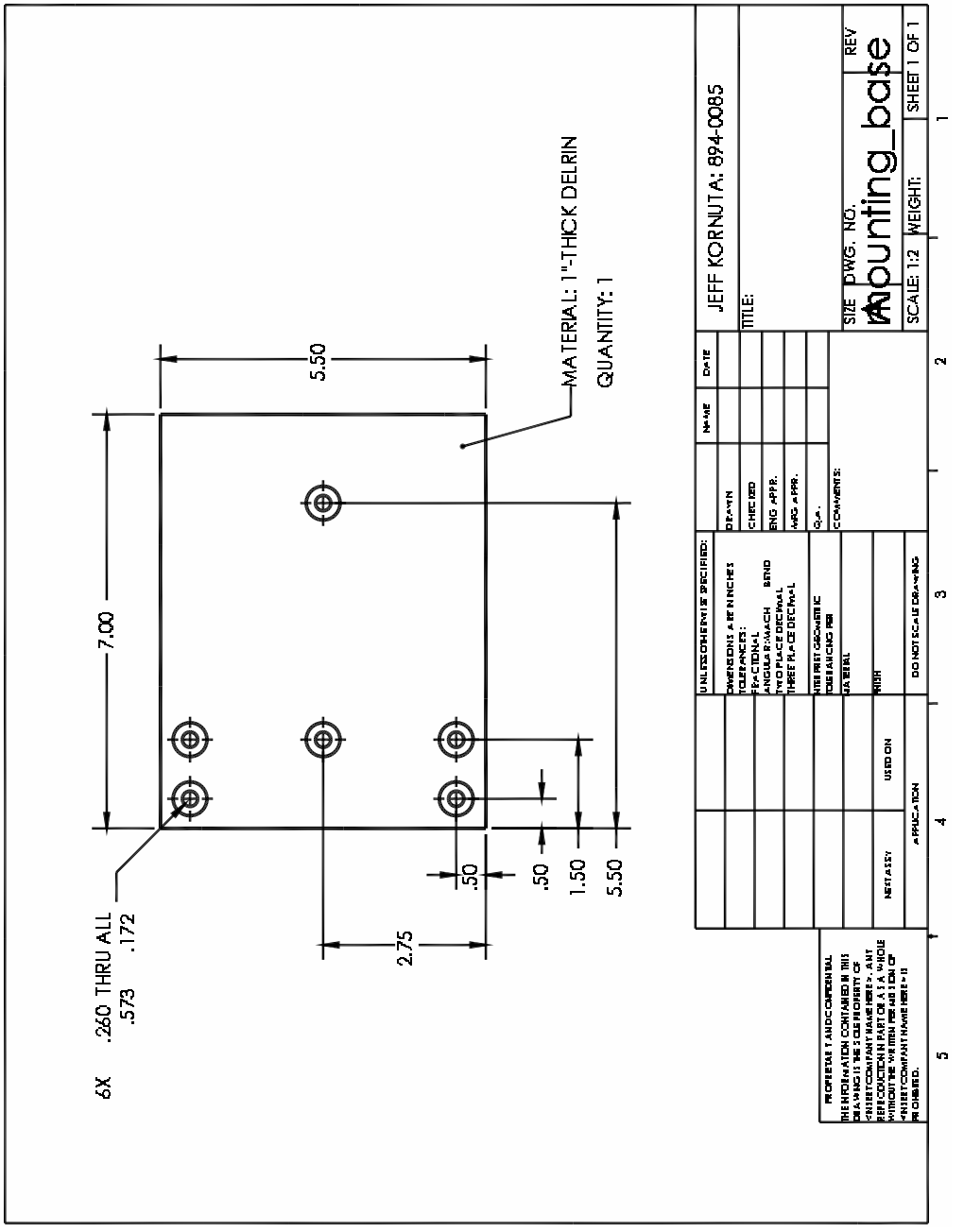
```
// strain_arduino_code.pde

// Include lcd library, EEPROM library, & stdio
//  => For use with standard 16x2 LCD
#include <LiquidCrystal.h>

// Connections:
//  rs (LCD pin 4) to Arduino pin 12
//  rw (LCD pin 5) to Arduino pin 11
//  enable (LCD pin 6) to Arduino pin 10
//  LCD pin 15 to Arduino pin 13
//  LCD pins d4, d5, d6, d7 to Arduino pins 5, 4, 3, 6
LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 6);

// Declare all constants
int backLight = 13;    // pin 13 will control the backlight
unsigned long timeOld = 0;
float rpm = 0.0;
int count = 0;

// Setup function
void setup()
{
  // Set backlight
  pinMode(backLight, OUTPUT);
  digitalWrite(backLight, HIGH); // turn backlight on. Replace 'HIGH
     ' with 'LOW' to turn it off.
  lcd.begin(16,2);              // columns, rows.  use 16,2 for a 16
     x2 LCD, etc.
  lcd.clear();                  // start with a blank screen
  lcd.setCursor(0,0);
  lcd.print("RPM: ");

  // Attach hardware interrupt to detect a falling voltage signal
     from photo
  // interrupter
  attachInterrupt(0, increment, FALLING);
}
```

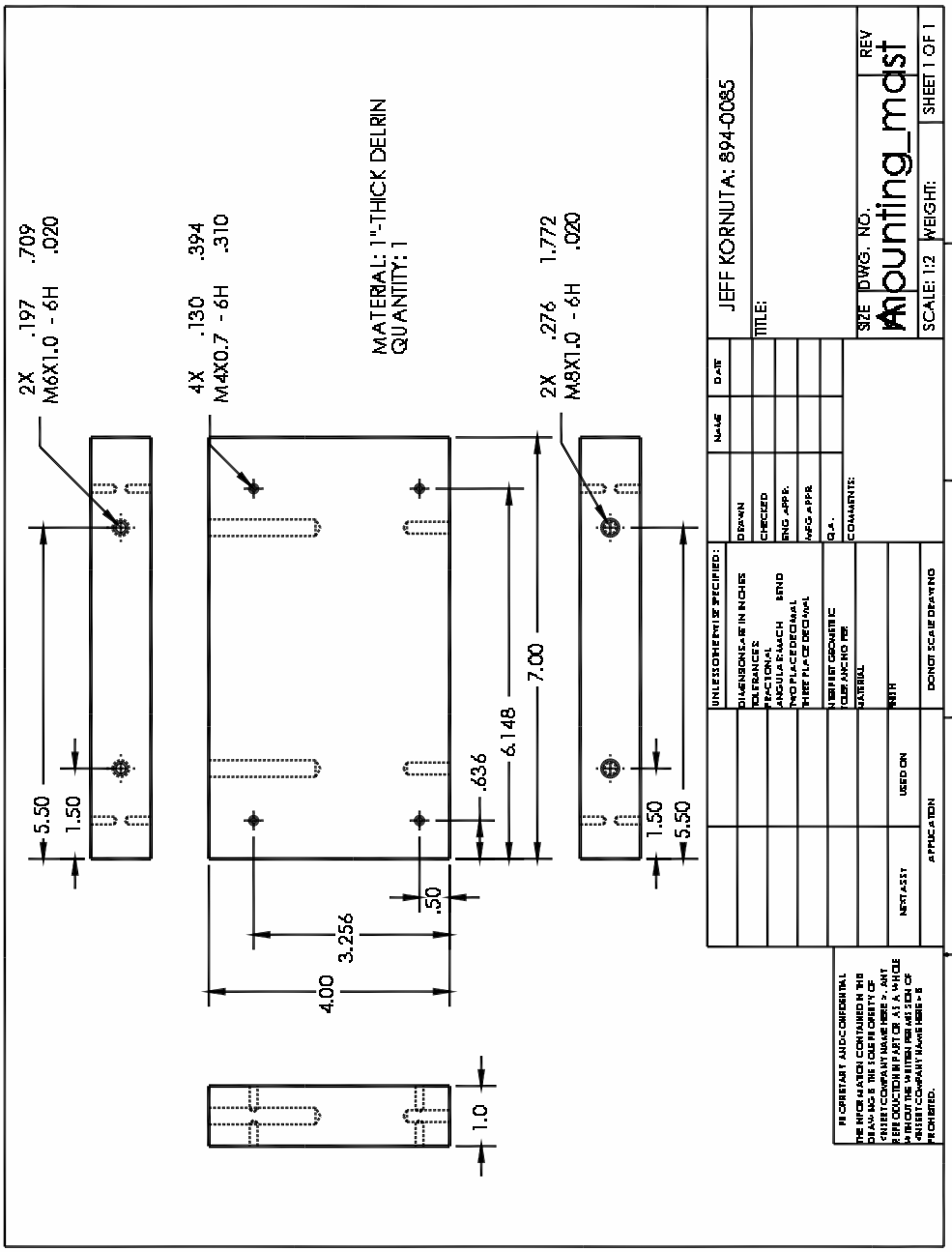
```
// Start main loop to measure RPM
void loop()
{
  // After three measurements, average then output on LCD
  if (count >= 3) {
    // Calculate RPM
    rpm = 60.0*1000.0/((float)millis() - (float)timeOld)/4.0;
    timeOld = millis();
    count = 0;

    // Output to LCD
    lcd.setCursor(0,1);
    lcd.print("                ");
    lcd.setCursor(4,1);
    lcd.print(rpm);
  }
}

// Function that interrupt is attached to
void increment()
{
  count++;
}
```

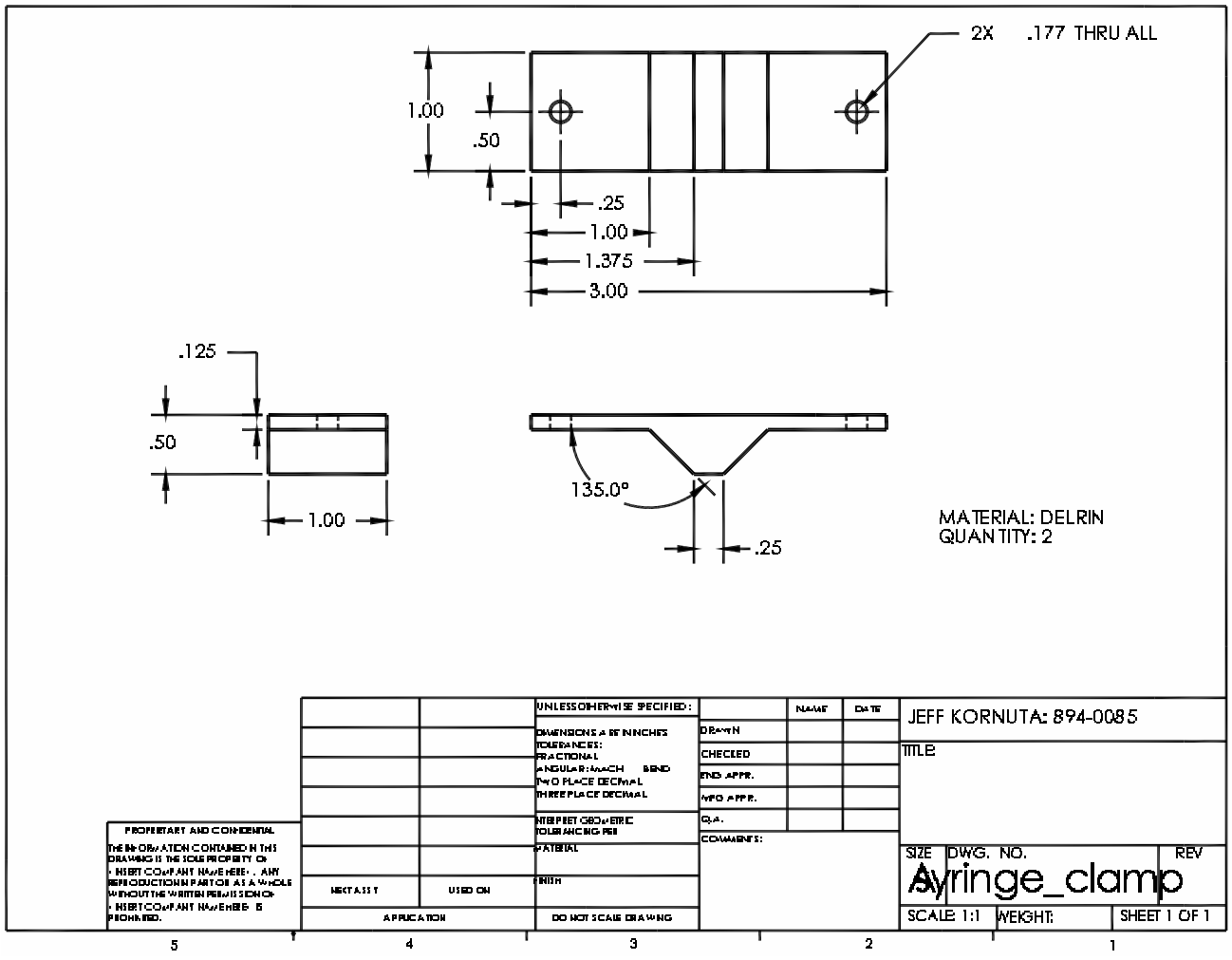
APPENDIX B

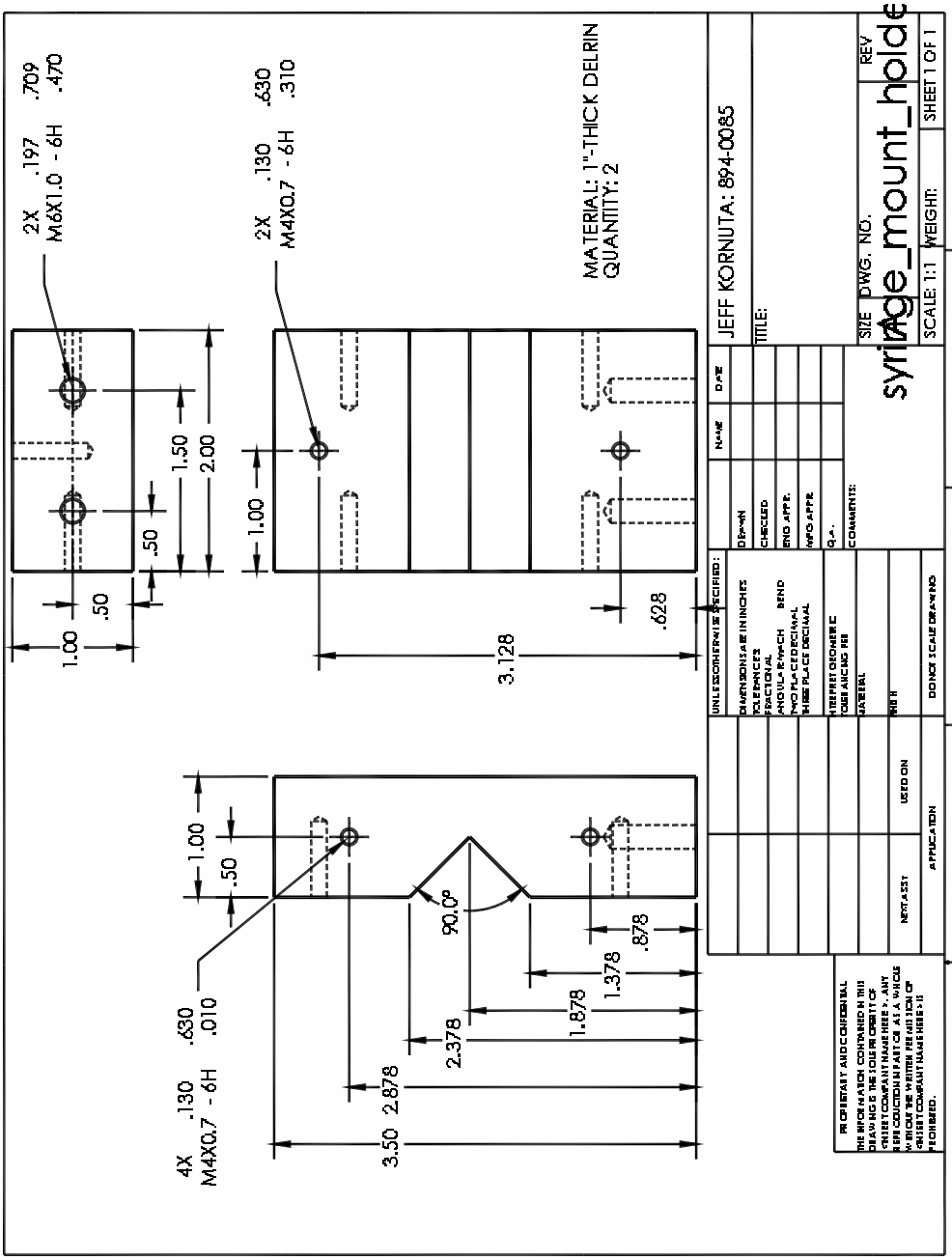
ANCILLARY MATERIAL FOR CHAPTER III

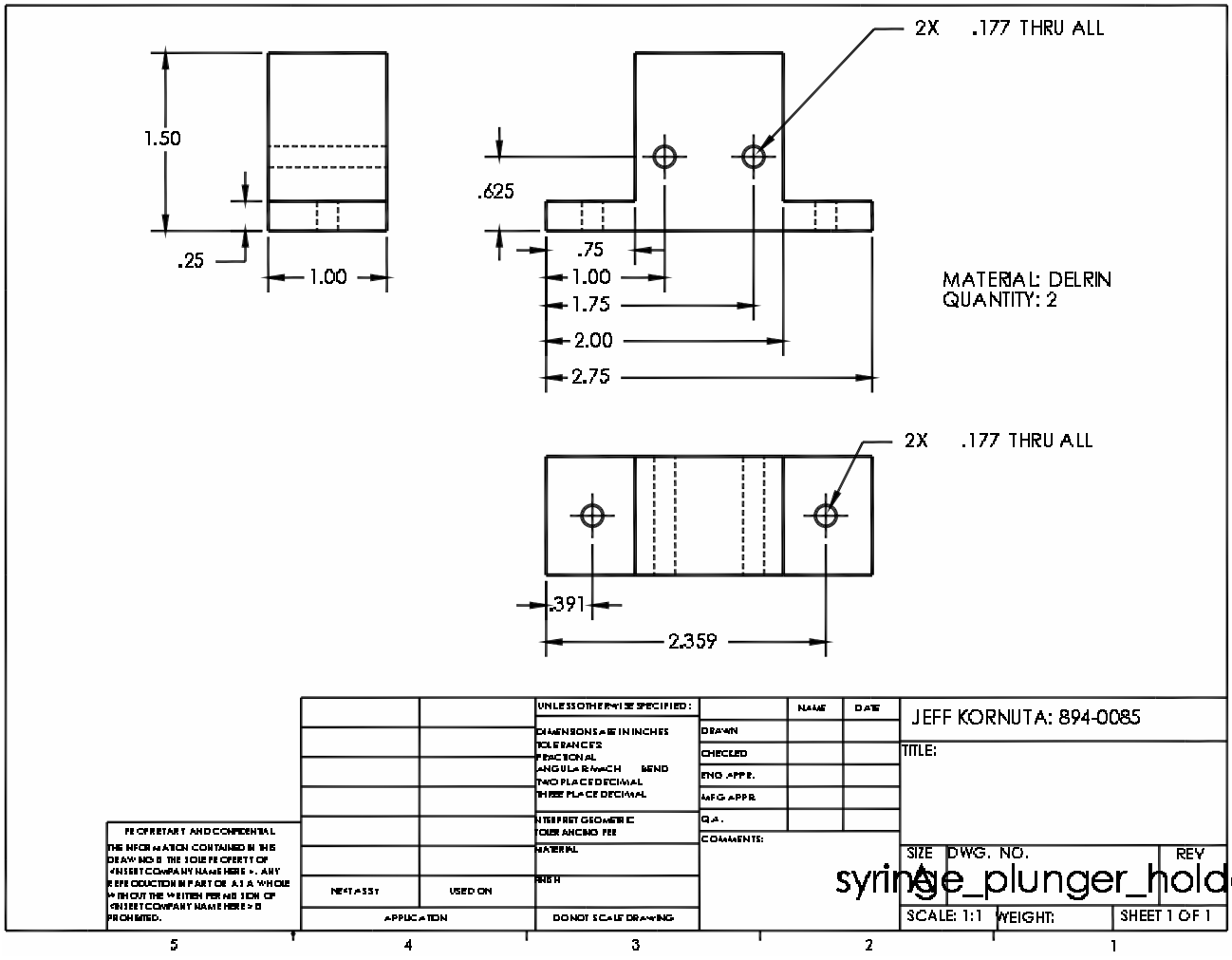
B.1 CAD Drawings for ELPS Hardware




2X .197 .709
M6X1.0 - 6H .020
4X .130 .394
M4X0.7 - 6H .310
MATERIAL: 1"-THICK DELRIN
QUANTITY: 1
2X .276 1.772
M8X1.0 - 6H .020
5.50
1.50
.636
6.148
7.00
.50
3.256
4.00
1.0
1.50
5.50
JEFF KORNUTA: 894-0085
TITLE:
Mounting_mast
SIZE
DWG. NO.
REV
SCALE: 1:2
WEIGHT:
SHEET 1 OF 1
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
DATE
NAME
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL
ANGULAR: MACH BEND
TWO PLACE DECIMAL
THREE PLACE DECIMAL
INTERPRET GEOMETRIC
TOLERANCING PER:
MATERIAL
FINISH
DO NOT SCALE DRAWING
NEXT ASSY
USED ON
APPLICATION
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS
DRAWING IS THE SOLE PROPERTY OF
<INSERT COMPANY NAME HERE>. ANY
REPRODUCTION IN PART OR AS A WHOLE
WITHOUT THE WRITTEN PERMISSION OF
<INSERT COMPANY NAME HERE> IS
PROHIBITED.

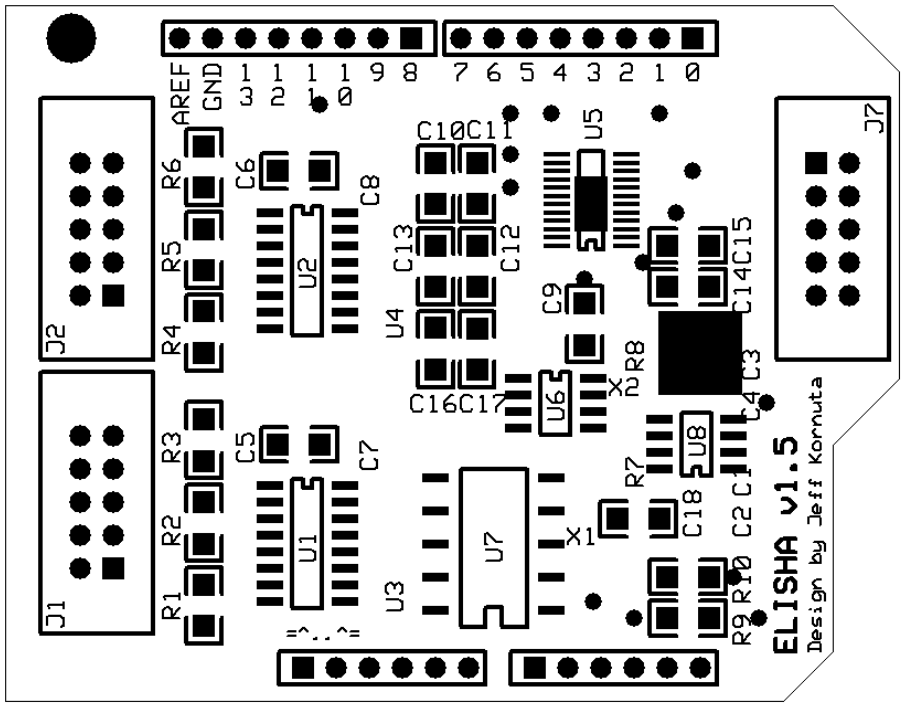

2X .177 THRU ALL
1.00
.50
.25
1.00
1.375
3.00
.125
.50
1.00
135.0°
.25
MATERIAL: DELRIN
QUANTITY: 2
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL
ANGULAR: MACH BEND
TWO PLACE DECIMAL
THREE PLACE DECIMAL
INTERPRET GEOMETRIC TOLERANCING PER
MATERIAL
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
JEFF KORNUTA: 894-0085
TITLE:
SIZE
DWG. NO.
REV
Ayringe_clamp
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
5
4
3
2
1

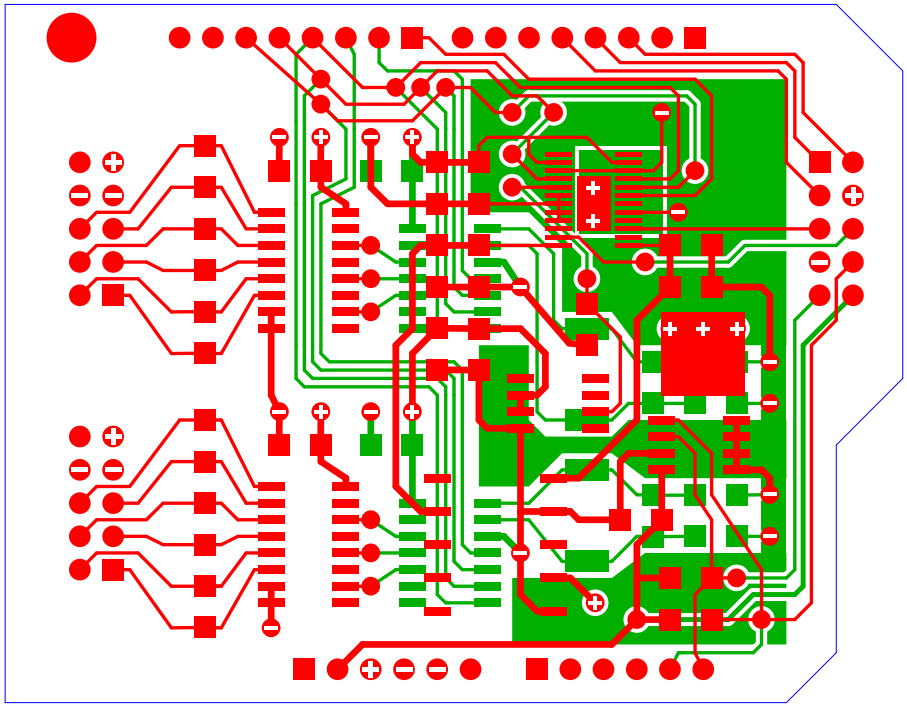

2X .197 .709
M6X1.0 - 6H .470
2X .130 .630
M4X0.7 - 6H .310
MATERIAL: 1"-THICK DELRIN
QUANTITY: 2
1.00
.50
1.50
2.00
3.128
.628
4X .130 .630
M4X0.7 - 6H .010
90.0°
.878
1.378
1.878
2.378
2.878
3.50
JEFF KORNUTA: 894-0085
syringe_mount_holde
SIZE
DWG. NO.
REV
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1

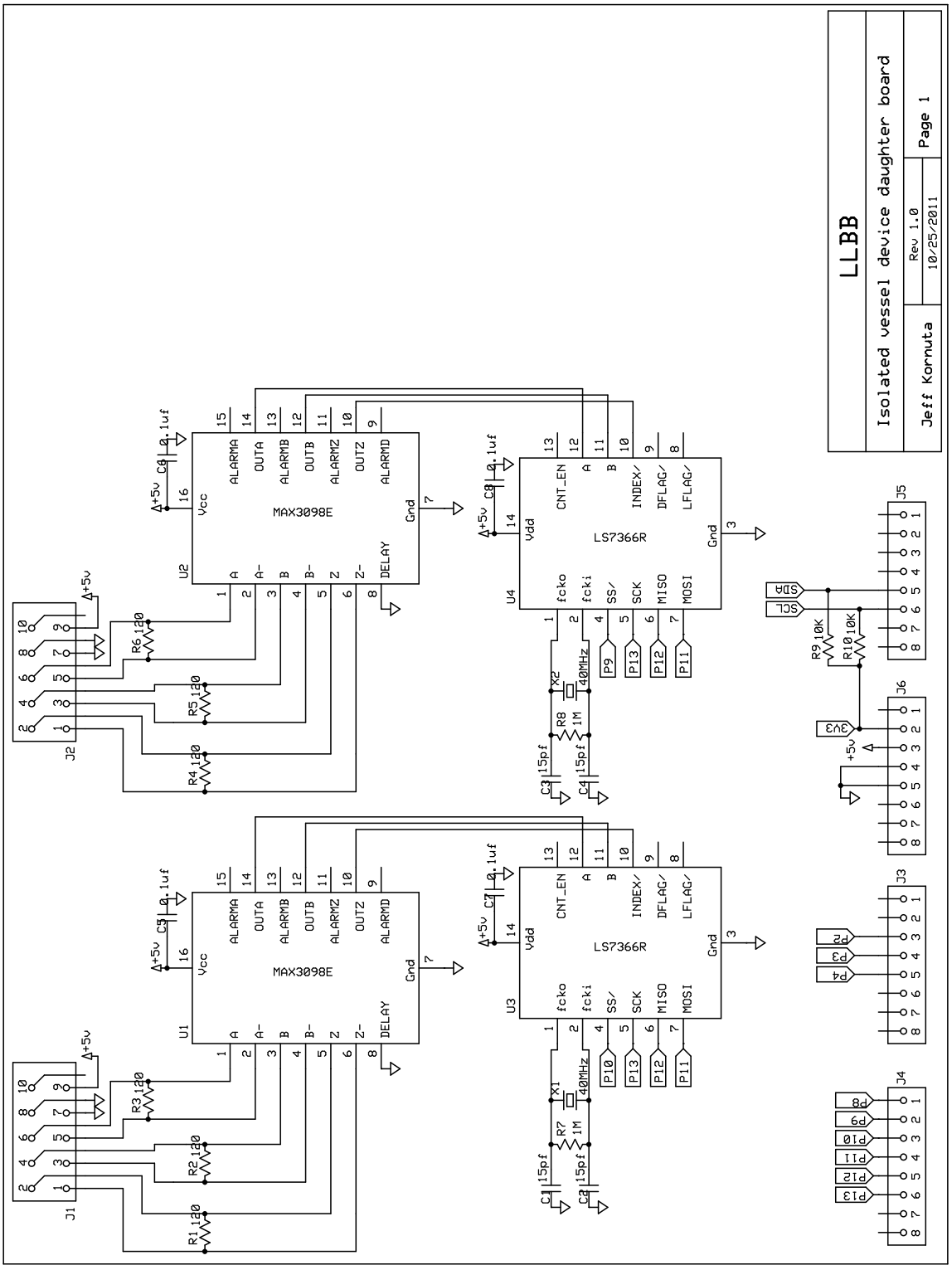

2X .177 THRU ALL
1.50
.25
1.00
.625
.75
1.00
1.75
2.00
2.75
MATERIAL: DELRIN
QUANTITY: 2
2X .177 THRU ALL
.391
2.359
UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL
ANGULAR: MACH BEND
TWO PLACE DECIMAL
THREE PLACE DECIMAL
INTERPRET GEOMETRIC TOLERANCING PER
MATERIAL
FINISH
NEXT ASSY
USED ON
APPLICATION
DO NOT SCALE DRAWING
NAME
DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
JEFF KORNUTA: 894-0085
TITLE:
SIZE
DWG. NO.
REV
A
syringe_plunger_holde
SCALE: 1:1
WEIGHT:
SHEET 1 OF 1
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.
5
4
3
2
1

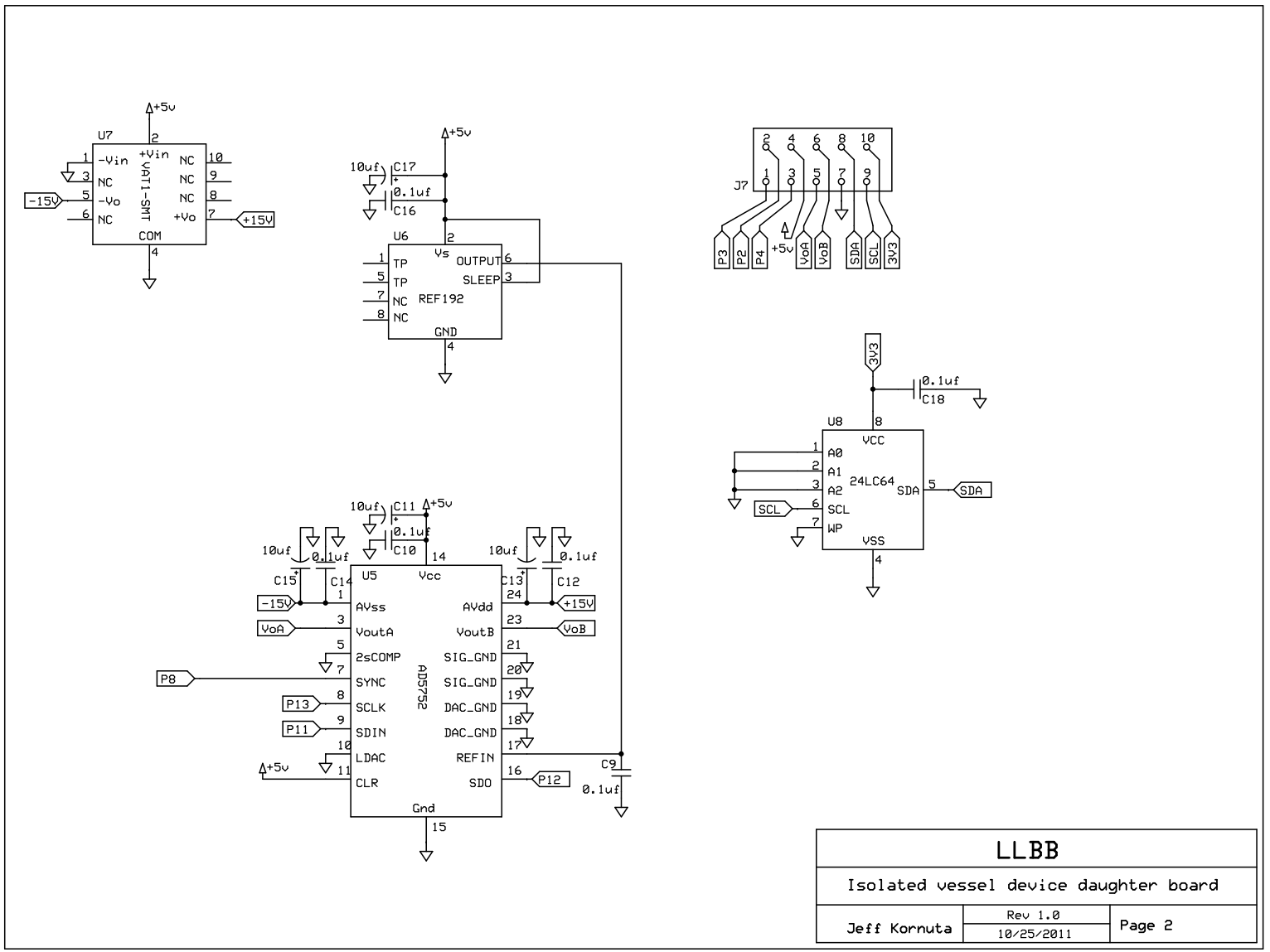
B.2 Daughter Board Layout and Schematics

B.2.1 ELectronic Interfacing System Hub and Actuator (ELISHA)





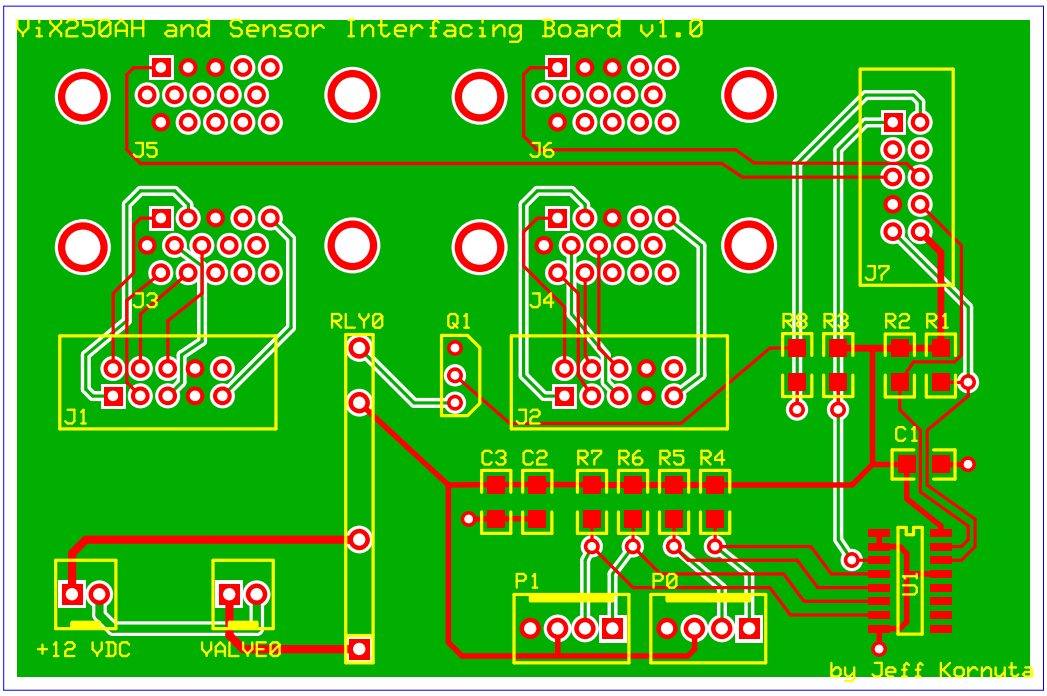

LLBB
Isolated vessel device daughter board
Jeff Kornuta
Rev 1.0
10/25/2011
Page 1
MAX3098E
LS7366R
U1
U2
U3
U4
J1
J2
J3
J4
J5
J6
X1
X2
40MHz
+5v
3V3
SDA
SCL


U7
+5v
-Vin
+Vin
NC
NC
NC
NC
-Vo
+Vo
COM
VA11-SMT
-15V
+15V
10uf
C17
0.1uf
C16
U6
Vs
TP
TP
NC
NC
OUTPUT
SLEEP
REF192
GND
J7
P3
P2
P4
+5v
VoA
VoB
SDA
SCL
3V3
U8
VCC
A0
A1
A2
SCL
WP
VSS
24LC64
SDA
0.1uf
C18
C11
C10
C15
C14
C13
C12
U5
Vcc
AVss
AVdd
VoutA
VoutB
2sCOMP
SYNC
SCLK
SDIN
LDAC
CLR
SIG_GND
DAC_GND
REFIN
SDO
Gnd
AD5752
P8
P13
P11
P12
C9
LLBB
Isolated vessel device daughter board
Jeff Kornuta
Rev 1.0
10/25/2011
Page 2

B.2.1.1 ELISHA BOM

Label	Component
C1–C4	15 pF
C5–C10, C12, C14, C16, C18	0.1 uF
C11, C13, C15, C17	10 uF
J1, J2, J7	2x5 Shrouded header
J4–J6	Male header
R1–R6	120 Ω
R7, R8	1 MΩ
R9, R10	10 kΩ
U1, U2	MAX3098E
U3, U4	LS7366R
U5	AD5752
U6	REF192
U7	VAT1-SMT
U8	24LC64 (optional)
X1, X2	40 MHz Crystal

B.2.2 ViX/Sensor Interfacing Board

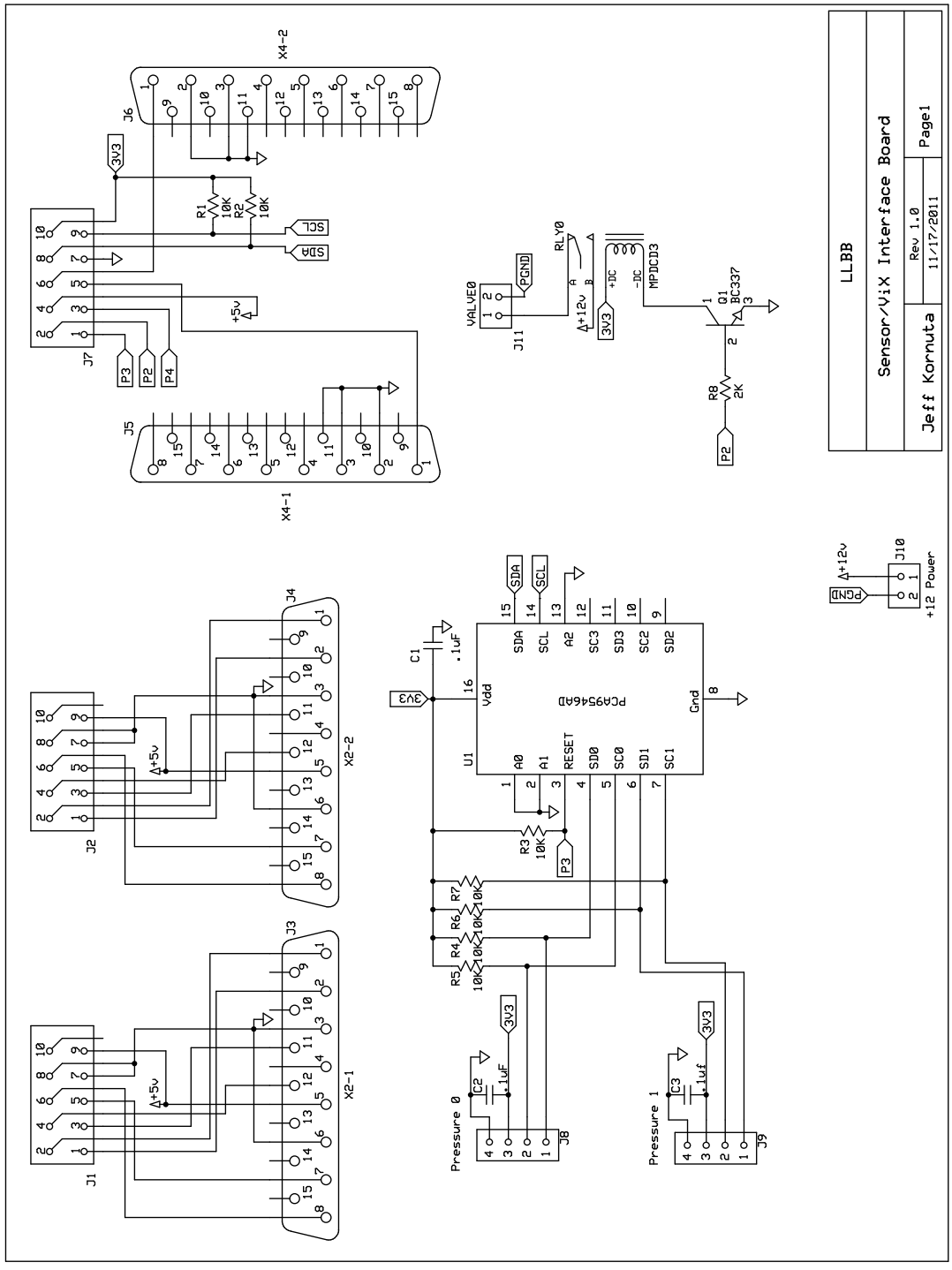


Known bugs:

1. HD-15 connectors (J3–J6) are spaced a bit too closely together.

2. HD-15 pins are spaced (J3–J6) a bit too tightly.

3. Through-hole diameters for relay pins (RLY0) are slightly small.


LLBB
Sensor/ViX Interface Board
Jeff Kornuta
Rev 1.0
11/17/2011
Page1
X4-2
X4-1
X2-2
X2-1
J1
J2
J3
J4
J5
J6
J7
J8
J9
J10
J11
U1
PCA9546AD
Vdd
Gnd
SDA
SCL
A0
A1
A2
RESET
SD0
SC0
SD1
SC1
SD2
SC2
SD3
SC3
C1
.1uF
C2
1uF
C3
1uf
R1
10K
R2
10K
R3
10K
R4
10K
R5
10K
R6
10K
R7
10K
R8
2K
3V3
+5v
+12v
PGND
P2
P3
P4
VALVE0
RLY0
MPDCD3
Q1
BC337
+12 Power
Pressure 0
Pressure 1

B.2.2.1 ViX/Sensor Interfacing Board BOM

Label	Component
C1–C3	0.1 uF
J1, J2, J7	2x5 Shrouded header
J3–J6	HD-15 Female connector
J8, J9	To pressure sensors
J10	+12 VDC
J11	To solenoid valve
Q1	BC337 transistor
R1–R7	10 kΩ
R8	2 kΩ
RLY0	MPDCD3 solid-state relay
U1	PCA9546AD

B.3 Sending Serial Commands to the ELPS

```
"""
  masterSerial.py

  Python program to interface serially with EVPS

  Written by Jeff Kornuta, 4/5/13
"""

import serial, time, sys, struct, scipy.io, os
from numpy import *

#####################################
# Global variables and data loading  #
#####################################
### Specify path of files
path = '../../Spring 2013/'

### For identification (mode 1), fill up appropriate array
data = scipy.io.loadmat(path + 'System Inputs/idinput-20130612
    _rbs300_30Hz_15sec.mat')
# Inputs stored in variable "u" when saved in MATLAB
u = data['u'].astype(float32)
# Data length
data_length_mode1 = u.size/2
# Initialize storage arrays
u1_string_mode1 = [ ]
u2_string_mode1 = [ ]
# Optional multipliers for signals 1 and 2
multiplier1 = 0.6
multiplier2 = 0.6
# Load up input values in string array
for i in xrange(data_length_mode1):
  u1_string_mode1.append( struct.pack('f', multiplier1*u[i,0]) )
  u2_string_mode1.append( struct.pack('f', multiplier2*u[i,1]) )


### For control (mode 2), fill up appropriate array
data = scipy.io.loadmat(path + 'System Inputs/
    end_exp_Ca_free_20130827.mat')
#data = scipy.io.loadmat(path + 'System Inputs/experiment_A_20130826
    .mat')
#data = scipy.io.loadmat(path + 'System Inputs/
    experiment_A_extra_steps_20130904.mat')
#data = scipy.io.loadmat(path + 'System Inputs/experiment_B_20130905
    .mat')
#data = scipy.io.loadmat(path + 'System Inputs/
    characterize_vessel_20130903.mat')
# Inputs stored in variable "u" when saved in MATLAB
u = data['u'].astype(float32)
#Ts = 6.25143e-3; # calibrated sampling time limited by my laptop
#t = arange(0.,12*60.,Ts)
#u1 = sin(2*pi*0.5*t) + 2.0
```

```
#u2 = sin(2*pi*0.75*t) - cos(2*pi*0.5*t) + 3.0
#u1 = sin(2*pi*0.7*t) - cos(2*pi*0.5*t) + 2.0
#u1 = 0*t + 0.0
#u2 = 0*t + 5.0
#u = array([u1, u2]).T
#u = u.astype(float32)
# Data length
data_length_mode2 = u.size/2
# Initialize storage arrays
u1_string_mode2 = [ ]
u2_string_mode2 = [ ]
# Optional multipliers for signals 1 and 2
multiplier1 = 1.0
multiplier2 = 1.0
# Load up input values in string array
for i in xrange(data_length_mode2):
  u1_string_mode2.append( struct.pack('f', multiplier1*u[i,0]) )
  u2_string_mode2.append( struct.pack('f', multiplier2*u[i,1]) )


#**********************#
# Start main function  #
#**********************#
def main():

  try:
    # Clear screen
    os.system( [ 'clear', 'cls' ][ os.name == 'nt' ] )

    # Create serial object
    BAUDRATE = 921600 # 115200 if slow
    #
    # ...For Windows
    #ser = serial.Serial()
    #ser.port = 11
    #ser.baudrate = BAUDRATE

    # ...For Mac OSX
    uno32 = serial.Serial('/dev/tty.usbserial-AE00DRYI', BAUDRATE)

    # Open serial connection
    uno32.timeout = 5
    uno32.open()
    uno32.isOpen()


    # Wait initially for connection to establish
    print "Connecting to device...\n"
    time.sleep(7)

    # Print output/status from system
    while uno32.inWaiting() > 0:
      line = uno32.readline()
      sys.stdout.write(line)
```

```
    # Start prompt
    prompt(uno32)

  except KeyboardInterrupt:
    print "Bye!"


#######################
# Start the prompt    #
#######################
def prompt(uno32):
  # Give summary of available commands to user
  print "Type 'help' to see list of commands or 'exit' to quit."

  # Wait for user to decide what to do.
  while (1):
    # Get user input
    input = raw_input(">> ")

    # Break if receive exit
    if input == 'exit':
      break
    # Start debug sequence
    elif input == 'debug':
      uno32.write('d')
      go_serial(uno32, 0.5) # Mode 0.5
      continue
     # Start debug sequence (write debug file)
    elif input == 'debugf':
      uno32.write('d')
      go_serial(uno32, 0) # Mode 0
      continue
    # Start identification
    elif input == 'ident':
      uno32.write('i')
      go_serial(uno32, 1) # Mode 1
      continue
    elif input == 'start':
      uno32.write('c')
      go_serial(uno32, 2) # Mode 2
      continue
    # Send size of data
    elif input == 'help':
      print """
      List of commands:
        'debug'  - Enter debug mode
        'debugf' - Enter debug mode (saving output to file)
        'ident'  - Start system identification
        'start'  - Start control experiment

      Life advice -- if in a bind, 'Ctrl+C'!
      """
      continue
```

```
    # Otherwise, wait for another command
    else:
      print "Unrecognized command.\n"
      continue

  # OK, out of while loop -- exit!
  uno32.close()
  os._exit(1)


#*****************************************************#
# Function to continuously R/W serial data for ID/other #
#*****************************************************#
def go_serial(uno32, mode):
  # DEBUG mode (mode 0 or 0.5)
  if mode == 0:
    # File to store incoming serial data (file name is current time
       stamp)
    filename = time.strftime("debug_%Y%m%d%H%M%S.txt")
    filename = path + 'Experiments/' + filename   # For all other
       experiments
    print "> Starting debug mode...\n"
  if mode == 0.5:
    # Send to the magical and philosophically complex /dev/null
    filename = os.devnull
    print "> Starting debug mode...\n"

  # Identification mode
  if mode == 1:
    # Send and verify length of data
    data_length = data_length_mode1
    check_data(uno32, data_length)
    # File to store incoming serial data (file name is current time
       stamp)
    filename = time.strftime("data_%Y%m%d%H%M%S.txt")
    filename = path + 'SysID Data/' + filename   # For SysID
       experiments
    # Set array variables
    u1_string = u1_string_mode1
    u2_string = u2_string_mode1
    print "> Starting identification...\n"

  # Control mode
  if mode == 2:
    # Send and verify length of data
    data_length = data_length_mode2
    check_data(uno32, data_length)
    # File to store incoming serial data (file name is current time
       stamp)
    filename = time.strftime("data_%Y%m%d%H%M%S.txt")
    filename = path + 'Experiments/' + filename   # For all other
       experiments
    # Send over PI gains
    #uno32.write( struct.pack('f', 0.0) ) # Kp gain 0.037
```

```
  #uno32.write( struct.pack('f', 0.0) ) # Ki gain 0.42
  # Set array variables
  u1_string = u1_string_mode2
  u2_string = u2_string_mode2
  print "> Starting...\n"

# Open file, log time
file = open(filename,'w')
t = time.time()

# Send first Hp+1 inputs to serial buffer
Hp = 5
if mode == 0 or mode == 0.5: # Mode 0
  for i in xrange(Hp+1):
    uno32.write(  '\x00\x00\x00\x00' ) # Dummy 0.0
    uno32.write(  '\x00\x00\x00\x00' ) # Dummy 0.0
else: # Modes 1 or 2
  for i in xrange(Hp+1):
    uno32.write( u1_string[i] )
    uno32.write( u2_string[i] )

# Start data counter (offset by data already sent)
data_counter = Hp+1

# Grab data!
while (1):
  # Send, grab and store data, but stop/exit if user hits Ctrl-C
  try:
    # Read in line of data, print, and write to file
    line = uno32.readline()
    # Once received a line of data, send next data immediately
    if mode == 0 or mode == 0.5: # Mode 0
      uno32.write(  '\x00\x00\x00\x00' ) # Dummy 0.0
      uno32.write(  '\x00\x00\x00\x00' ) # Dummy 0.0
    else: # Modes 1 or 2
      if data_counter < data_length:
        uno32.write( u1_string[data_counter] )
        uno32.write( u2_string[data_counter] )
      else:   # For MPC prediction horizon, just send last data
         point until done
        uno32.write( u1_string[data_length-1] )
        uno32.write( u2_string[data_length-1] )


    # Increment counter
    data_counter = data_counter + 1

    # Get time; output data to screen and write to file
    elapsed = time.time() - t
    line = "%.3f  " % elapsed + line
    sys.stdout.write(line)
    file.write(line)

    # Close if done (modes 1 or 2)
```

```
      if mode == 1 or mode == 2:
        if data_counter == data_length + (Hp):
          file.close()
          print "\n> Done. File closed. \n"
          elapsed = time.time() - t
          print "\n> Elapsed time: " + str(elapsed) + " sec\n"
          print "\n> Exiting...\n"
          # Return to prompt
          #prompt(uno32)
          # Exit program
          uno32.close()
          os._exit(1)

    # Ctrl-C gracefully exits loop
    except (KeyboardInterrupt, SystemExit):
      file.close()
      print "\n> File closed. Exiting...\n"
      # Return to prompt
      #prompt(uno32)
      # Exit program
      uno32.close()
      os._exit(1)


#*****************************************#
# Function to send input data to board       #
#*****************************************#
def check_data(uno32, data_length):
  # Send amount of data to be sent (string-packed)
  uno32.write( struct.pack('I',data_length) )

  # Check data integrity
  line = uno32.readline()
  # Was data send successfully?
  if line == (str(data_length)+'\r\n'):
    print "> Data sent successfully.\n"
  else:
    print "> ERROR: data transfer unsuccessful. Try again.\n"
    prompt(uno32)


#*****************#
# Pythonic stuff  #
#*****************#

# Run main() if executed from command line
if __name__ == "__main__":
  main()
```

B.4 ELPS Software

```
/*
```

```
 *  Master_1.pde
 *
 *  MASTER CONTROL PROGRAM
 *  Debug, System Identification, or Controlling EVPS using PID/MPC.
 *    JK 4/5/2013
 *
 */

#if defined(__PIC32MX__)
#include <p32xxxx.h>     /* this gives all the CPU/hardware
   definitions */
#include <plib.h>        /* this gives the i/o definitions */
#endif

#include <Wire.h>
#include "HSC.h"
#include "Servo.h"
#include "SPI.h"
#include <math.h>
#include <stdbool.h>
#include <stdint.h>
#include "Controller.h" // for special controller methods

/********************
* Define Statements *
********************/

#define DAC_A  0
#define DAC_B  2

#define CLOCK_FREQ 80000000
#define TIMER_PRESCALE 4
#define PRESCALED_TIMER_FREQ 20000000
#define PI 3.14159265

/************************
* Variable Declarations *
************************/
// Servo objects
Servo servo1 (10, 8, DAC_A); // QD cs = 10, DAC cs = 8, DAC A
Servo servo2 (9, 8, DAC_B);

// Pressure sensor objects
HSC pSensor1 (1);   // Pressure sensor #1
HSC pSensor2 (2);   // Pressure sensor #2

// Controller object
Controller control;

// For filtering
float p1_history[max_samples] = {0};// Initialize pressure histories
float p2_history[max_samples] = {0};
uint8_t filter_order = 12;
// For PID (if using)
```

```
float Kp;  // P-gain for Pavg
float Ki;  // I-gain for Pavg
float p1_error_sum = 0.0;// Sum of error, p1
float p2_error_sum = 0.0;// Sum of error, p2
// For MPC (if using)
float xhat[n][1] = {0}; // Initialize state estimate, xhat (xhat_0)
float Ydk[Hp*p][1] = {0}; // Initialize input (desired output)
   vector
float y[p][1] = {0};  // Initialize output vector
float u[m][1] = {0};  // Initialize input vector
// For solenoid switching
bool xend1 = false; // If motor 1 reaches end
bool xend2 = false; // If motor 2 reaches end
bool xmid1 = true;  // If motor 1 crosses midway
bool xmid2 = true;  // If motor 2 crosses midway
bool switched = false; // Solenoid switch state
uint8_t solenoid = 2; // Solenoid switching pin
float x1_old = 0;
float x2_old = 0;
// Other variables
uint8_t mode; // Program mode: 0 - debug, 1 - identification, 2 -
   control
uint32_t data_length = 0;    // Length of input data
uint32_t data_counter = 0;
char buffer [80]; // String buffer to serially print back to screen
bool apply_value = false;
bool pin_state = false;
double delta_t;
//double Ts = 0.005;      // Sampling time, sec
//double Fs = 1.0 / Ts;   // Sampling frequency, Hz
double Fs = 300.0;
double Ts = 1.0 / Fs;


/************************
 * Function Declarations *
 ************************/
void configureTimer45();
void serialWaitBytes();
uint32_t readULongFromBytes();
float readFloatFromBytes();

/************************
 * Function Definitions *
 ************************/

void setup()
{
  // Start serial and I2C communication
  // http://www.chipkit.org/forum/viewtopic.php?f=19&t=711
  Serial.begin(921600);
  Wire.begin();

  // Set solenoid pin as output (start out LOW)
```

```
pinMode(solenoid, OUTPUT);
digitalWrite(solenoid, switched);

// Wait for devices to warm up
delay(100);

// Set up SPI and configure QD and DAC
servo1.init();
servo2.init();

// Initial output voltage, 0 V
servo1.move(0.0);
servo2.move(0.0);

// Wait until mode specified, then begin
bool wait = true;
while ( wait )
{
  if ( Serial.available() > 0 )
  {
    // Read in byte
    char in = Serial.read();

    // Was 'd' received? (i.e. for debug)
    if ( in == 'd' )
    {
      // Specify mode 0
      mode = 0;

      // Exit out of wait loop
      wait = false;
    }

    // Was 'i' received? (i.e. for identification)
    if ( in == 'i' )
    {
      // Specify mode 1
      mode = 1;

      // Great, read in length of input data
      data_length = readULongFromBytes();

      // Print to verify data integrity
      Serial.println( data_length,DEC );

      // Exit out of wait loop
      wait = false;
    }

    // Was 'c' received? (i.e. for control)
    if ( in == 'c' )
    {
      // Specify mode 2
      mode = 2;
```

```
      // Great, read in length of input data
      data_length = readULongFromBytes();

      // Print to verify data integrity
      Serial.println( data_length,DEC );

      /* For PID
      // Read in control gains
      Kp = readFloatFromBytes();
      Ki = readFloatFromBytes();
      */

      // For MPC
      // Fill up initial Ydk vector
      for (uint8_t i = 0; i < Hp*p; i = i+p)
      {
        // Read from serial buffer
        Ydk[i][0] = readFloatFromBytes();
        Ydk[i+1][0] = readFloatFromBytes();
      }

      // Estimate current xhat (from current output) as initial
         state!
      float p1 = pSensor1.pressure();
      float p2 = pSensor2.pressure();
      float y1 = p1-p2;
      float y2 = (p1+p2)/2.0;
      y[0][0] = y1;
      y[1][0] = y2;
      control.stateInit(y, xhat);

      // Use current pressure to fill up pressure history...
      // ...will make the filtering start buttery-smooth
      for (uint8_t i = 0; i <= filter_order; i++)
      {
        p1_history[i] = p1;
        p2_history[i] = p2;
      }

      // Exit out of wait loop
      wait = false;
    }
  }
}

// Zero position counters and pause to wait for serial buffer
//  to be filled, if need be
servo1.zero();
servo2.zero();
delay(3000); // Pause 3 sec

// Configure timer for desired sampling frequency
configureTimer45(Fs);
```

```
}

// Loops continuously
void loop()
{
  // If interrupt was triggered, do something!
  if (apply_value)
  {
    // If completed sending data, stop (for ID or control)
    if ( data_counter == data_length-1 && mode != 0 )
    {
      servo1.move(0.0);
      servo2.move(0.0);
      digitalWrite(solenoid, LOW);
      Serial.println( "Done!" );
      while(1){};
    }

    // Take in next desired input floats from serial buffer
    float r1 = readFloatFromBytes();
    float r2 = readFloatFromBytes();

    // Grab the motor positions; stop if out of range (+- xlimit mm)
    float x1 = servo1.position();
    float x2 = servo2.position();
    float xlimit = 30.0;
    if (abs(x1) > xlimit || abs(x2) > xlimit)
    {
      // Stop motors; end program
      servo1.move(0.0);
      servo2.move(0.0);
      digitalWrite(solenoid, LOW);
      Serial.println( "Motor(s) out of range!" );
      while(1){};
    }

    // Determine conditions for solenoid switching
    // End location at which solenoid should switch
    float xswitch = 15.0;
    // Logic portion: set proper boolean variables
    // If motor 1 reaches end
    if (abs(x1) > xswitch)
    {
      xend1 = true;
    }
    else
    {
      xend1 = false;
    }
    // If motor 2 reaches end
    if (abs(x2) > xswitch)
    {
      xend2 = true;
```

```
}
else
{
  xend2 = false;
}
// If motor 1 crosses midway (changes direction)
if (x1*x1_old < 0)
{
  xmid1 = true;
}
// If motor 2 crosses midway
if (x2*x2_old < 0)
{
  xmid2 = true;
}
// Store old position values
x1_old = x1;
x2_old = x2;

// One more failsafe...
if (abs(x1) > 20.0 || abs(x2) > 20.0)
{
  // Trick into thinking it's crossed midway
  xmid1 = true;
  xmid2 = true;
}

// Grab pressure values; stop if pressure out of range (+-
   plimit cmH2O)
float p1 = pSensor1.pressure();
float p2 = pSensor2.pressure();
float plimit = 60.0;
if (abs(p1) > plimit || abs(p2) > plimit)
{
  // Stop motors; end program
  servo1.move(0.0);
  servo2.move(0.0);
  digitalWrite(solenoid, LOW);
  Serial.println( "Pressure(s) out of range!" );
  while(1){};
}

// If mode 0 or 1, apply desired inputs, print serial buffer,
   continue
if ( mode == 0 || mode == 1 )
{
  // Apply inputs
  servo1.move(r1);
  servo2.move(r2);

  // Print serial buffer: [ r1 r2  x1 x2  p1 p2  time/dummy ]
  sprintf( buffer, "%.3f %.3f  %.3f %.3f  %.3f %.3f  %d",
    r1, r2, x1, x2, p1, p2, 0 );
  Serial.println(buffer);
```

```
}

// If mode 2, do control, print serial buffer, continue
if ( mode == 2 )
{
  // Moving average filter on pressure outputs
  p1 = control.filter(p1, p1_history, filter_order);
  p2 = control.filter(p2, p2_history, filter_order);
  float y1 = p1-p2;
  float y2 = (p1+p2)/2.0;

  // If conditions are right -- switch solenoid!
  if ((xend1 && xmid1) || (xend2 && xmid2))
  {
    // Change switched state
    switched = !switched;
    digitalWrite(solenoid, switched);

    // Turn off midway booleans
    xmid1 = false;
    xmid2 = false;
  }

  /* For PID
  // Calculate control inputs to system
  float p1_error = r1-y1;
  p1_error_sum += p1_error;
  float u1 = Kp*p1_error + Ki*Ts*p1_error_sum;

  float p2_error = r2-y2;
  p2_error_sum += p2_error;
  float u2 = Kp*p2_error + Ki*Ts*p2_error_sum;
  */

  // For MPC
  // Make updated desired input vector, Ydk
  for (uint8_t i = 0; i < (Hp-1)*p; i = i+p)
  {
    // Shift everything upwards
    Ydk[i][0] = Ydk[i+p][0];
    Ydk[i+1][0] = Ydk[i+p+1][0];
  }
  Ydk[Hp*p-2][0] = r1;  // Fill in last spots with most recent
     serial grab
  Ydk[Hp*p-1][0] = r2;
  // Calculate control inputs to system via MPC
  control.mpc(Ydk, xhat, u);
  float u1 = u[0][0];
  float u2 = u[1][0];

  // Apply control inputs
  //  (take solenoid direction into account with conditional)
  if (switched == false)
  {
```

```
        servo1.move( u1 );
        servo2.move( u2 );
      }
      else
      {
        servo1.move( u2 );
        servo2.move( u1 );
      }

      // For MPC
      // Estimate next state vector using estimator; update desired
         outputs
      y[0][0] = y1;
      y[1][0] = y2;
      control.stateEstim(u, y, xhat);
      r1 = Ydk[0][0];
      r2 = Ydk[1][0];

      // Print out stuff: [ r1 r2  y1 y2  u1 u2  x1 x2  p1 p2
         switched ]
      sprintf(buffer,
        "%.3f %.3f  %.3f %.3f  %.3f %.3f  %.3f %.3f  %.3f %.3f  %d",
        r1, r2, y1, y2, u1, u2, x1, x2, p1, p2, switched);
      Serial.println(buffer);
    }

    // Turn off apply_value (interrupt) trigger, increment counter
    apply_value = false;
    data_counter++;
  }

}

// Function to wait until number of bytes are in buffer
void serialWaitBytes(uint16_t num_bytes)
{
  while ( !(Serial.available() >= num_bytes) );
}

// Function to read in uint32_t from bytes
uint32_t readULongFromBytes()
{
  // Use union for C++ magic
  union u_tag
  {
    uint8_t b[4];
    uint32_t ulval;
  } u;

  // Read in bytes
  serialWaitBytes(4);
  for ( uint8_t i = 0; i < 4; i++)
  {
    u.b[i] = Serial.read();
```

```
  }

  // Return unsigned long
  return u.ulval;
}

// Function to read in 32-bit float from bytes
float readFloatFromBytes()
{
  // Use union for C++ magic
  union u_tag
  {
    uint8_t b[4];
    float fval;
  } u;

  // Read in bytes
  serialWaitBytes(4);
  for ( uint8_t i = 0; i < 4; i++)
  {
    u.b[i] = Serial.read();
  }

  // Return float
  return u.fval;
}


// Configure timer interrupt
void configureTimer45(double freq)
{
  uint32_t t_period;  //For the PIC32 PR4 register

  T4CON = 0x0;
  T5CON = 0x0;

  // Using the desired frequency, clock frequency,
  //  and number of vals per cycle, we can get the
  //  Timer period.
  t_period = (uint32_t) ((double) PRESCALED_TIMER_FREQ / freq );
  delta_t = (double) t_period * (1.0 / (double) PRESCALED_TIMER_FREQ
     );

  //T4CONSET = 0x18; //Prescaler 1:2, internal peripheral source
  T4CONSET = 0x28; //Prescaler 1:4, internal source

  TMR4 = 0;
  PR4 = t_period;
  IPC5SET = 0x5; //Priority level 1, sub-priority level 1
  IFS0CLR = 0x00100000;
  IEC0SET = 0x00100000;

  T4CONSET = 0x8000; // Start timer 45
```

```
}

#ifdef __cplusplus
extern "C" {
#endif
  void __ISR(_TIMER_5_VECTOR,IPL3AUTO) write_handler( void )
  {
    apply_value = true;
    IFS0CLR = 0x00100000;  // Clear interrupt flag
  }

#ifdef __cplusplus
}
#endif
```

```
/*
 *  Controller library for implementing control algorithms
 *    Uses methods from MatrixMath library by Charlie Matlack
 *
 *  By Jeff Kornuta on March 5, 2013
 */

#ifndef Controller_h
#define Controller_h

#include "WProgram.h"

/*********************
* Define Statements *
*********************/
#define rowsof(MATRIX) (sizeof (MATRIX) / sizeof *(MATRIX))
#define colsof(MATRIX) (sizeof *(MATRIX) / sizeof **(MATRIX))
#define matarg(MATRIX) rowsof(MATRIX), colsof(MATRIX), (MATRIX)
#define n 6 // Number of states
#define m 2 // Number of inputs
#define p 2 // Number of outputs
#define Hp 5 // Size of prediction horizon
#define max_samples 16 // Maximum size of sample history for filter

/*********************
* Class Declaration *
*********************/
class Controller
{
  public:
          Controller();
          void stateEstim(float u[m][1], float y[p][1], float xhat[n
             ][1]);
    void mpc(float Ydk[Hp*m][1], float xhat[n][1], float u[m][1]);
    void stateInit(float y[p][1], float xhat[n][1]);
    float filter(float y, float Y[max_samples], int order);
    float VectorNorm(float* Vector, int size);

    // Methods taken from MatrixMath library
```

```
    void MatrixCopy(float* A, int rowsA, int colsA, float* B);
          void MatrixMult(float* A, float* B, int rowsA, int colsA,
             int colsB, float* C);
          void MatrixAdd(float* A, float* B, int rowsA, int colsA,
             float* C);
        void MatrixSubtract(float* A, float* B, int rowsA, int colsA
           , float* C);
          void MatrixTranspose(float* A, int rowsA, int colsA, float
             * C);

};

#endif
```

```
/*
 *  Controller library for implementing control algorithms
 *    Uses methods from MatrixMath library by Charlie Matlack
 *
 *  By Jeff Kornuta on March 5, 2013
 */

#include "WProgram.h"
#include "Controller.h"
#include <math.h>

/***********************
* Function Definitions *
***********************/
// Constructor
Controller::Controller(void)
{
}

// Produce new state estimate, xhat_new
void Controller::stateEstim(float u[m][1], float y[p][1], float xhat
   [n][1])
{
  // Define system parameters and matrices
float G[n][n] = { 0.9881, -0.0141, 0.0380, -0.0263, 0.0585, -0.0346,
        0.0169, 0.9762, -0.1287, -0.0818, 0.1772, -0.1024,
        0.0144, 0.0479, 0.7663, -0.4201, -0.1078, -0.1773,
        -0.0391, 0.0528, 0.2442, 0.7782, 0.0895, -0.4801,
        -0.0635, -0.0826, 0.2312, -0.0363, 0.8000, 0.1224,
        0.0345, 0.1490, 0.2556, 0.2840, -0.3122, 0.6145 };

float H[n][m] = { -0.0001, 0.0001,
        0.0001, -0.0009,
        -0.0002, -0.0050,
        -0.0092, -0.0048,
        -0.0041, 0.0033,
        -0.0079, 0.0060 };

float C[p][n] = { 2.0579, -122.2165, 5.0683, -0.1801, -4.6082,
   -4.1844,
```

```
        218.3934, -11.3770, 0.2459, 0.9597, 2.7897, -3.7255 };

float L[n][p] = { -0.0013, 0.0042,
        -0.0116, -0.0014,
        -0.0084, 0.0261,
        0.0077, 0.0085,
        -0.0006, 0.0019,
        0.0166, 0.0026 };

  //
  // Prediction term
  //
  // Multiply G*xhat
  float Gxhat[n][1];
  MatrixMult((float*)G, (float*)xhat, n, n, 1, (float*)Gxhat);
  //multiply(matarg(G), matarg(xhat), Gxhat);

  // Multiply H*u
  float Hu[n][1];
  MatrixMult((float*)H, (float*)u, n, m, 1, (float*)Hu);
  //multiply(matarg(H), matarg(u), Hu);

  // Initial estimate of new state, xhat_new_est = Gxhat + Hu
  float xhat_new_est[n][1];
  MatrixAdd((float*)Gxhat, (float*)Hu, n, 1, (float*)xhat_new_est);
  //add(matarg(Gxhat), Hu, xhat_new_est);

  //
  // Correction term
  //
  // Multiply C*xhat
  float Cxhat[p][1];
  MatrixMult((float*)C, (float*)xhat, p, n, 1, (float*)Cxhat);
  //multiply(matarg(C), matarg(xhat), Cxhat);

  // Subtract Cxhat from y
  float y_minus_Cxhat[p][1];
  MatrixSubtract((float*)y, (float*)Cxhat, p, 1, (float*)
     y_minus_Cxhat);
  //subtract(matarg(y), Cxhat, y_minus_Cxhat);

  // Multiply L*(y_minus_Cxhat)
  float xhat_new_correction[n][1];
  MatrixMult((float*)L, (float*)y_minus_Cxhat, n, p, 1,
      (float*)xhat_new_correction);
  //multiply(matarg(L), matarg(y_minus_Cxhat), xhat_new_correction);

  //
  // New state estimation, xhat_new = xhat_new_est +
     xhat_new_correction
  //
  float xhat_new[n][1];
  MatrixAdd((float*)xhat_new_est, (float*)xhat_new_correction, n, 1,
      (float*)xhat_new);
```

```
  //add(matarg(xhat_new_est), xhat_new_correction, xhat_new);

  // Update previous state estimation with new state estimation
  MatrixCopy((float*)xhat_new, n, 1, (float*)xhat);

}

// Generate system input, u, using a model predictive controller (
   MPC)
void Controller::mpc(float Ydk[Hp*p][1], float xhat[n][1], float u[m
   ][1])
{
  // Define gain matrices
float Kca[Hp*p][n] = { 0.1950, -119.3514, 17.5090, 6.6544, -24.4752,
    8.4902,
        215.2618, -14.9064, 9.8812, -5.3278, 14.2234, -8.8409,
        0.0129, -112.0410, 26.9179, 10.8815, -44.6560, 8.1323,
        211.5896, -19.8849, 17.3999, -15.7643, 22.5574, -8.8056,
        1.1948, -102.6132, 29.4551, 10.2557, -60.0402, 1.0033,
        207.8653, -25.5676, 23.0491, -26.8325, 26.3719, -3.4501,
        3.3131, -93.1284, 24.6994, 6.4341, -68.7122, -6.4155,
        204.5460, -30.8939, 27.5152, -35.8735, 24.9274, 5.3302,
        5.9418, -84.7171, 15.0811, 2.8339, -71.3590, -10.4020,
        201.9889, -34.8829, 31.1987, -41.7164, 18.6021, 14.7577 };

float K1[m][Hp*p] = { 0.0171, -0.0039, -0.0244, 0.0309, -0.0031,
   0.0698, 0.0581, 0.0956, 0.1244, 0.1062,
        0.0221, 0.0045, -0.0198, -0.0004, -0.0857, 0.0293, -0.1343,
           0.0721, -0.1503, 0.1084 };

  // Multiply Kca*xhat
  float Kcaxhat[Hp*p][1];
  MatrixMult((float*)Kca, (float*)xhat, Hp*p, n, 1, (float*)Kcaxhat)
     ;
  //multiply(matarg(Kca), matarg(xhat), Kcaxhat);

  // Subtract Kcaxhat from Ydk
  float Ydk_minus_Kcaxhat[Hp*p][1];
  MatrixSubtract((float*)Ydk, (float*)Kcaxhat, Hp*p, 1,
      (float*)Ydk_minus_Kcaxhat);
  //subtract(matarg(Ydk), Kcaxhat, Ydk_minus_Kcaxhat);

  // Multiply K1*Ydk_minus_Kcaxhat ( = u )
  MatrixMult((float*)K1, (float*)Ydk_minus_Kcaxhat, m, Hp*p, 1, (
     float*)u);
  //multiply(matarg(K1), matarg(Ydk_minus_Kcaxhat), u);

  // Saturation checking: Servo library has built-in
  //  +-10 V saturation in move() method.
}

// Estimate initial condition, xhat(k=0), method
void Controller::stateInit(float y[p][1], float xhat[n][1])
{
```

```
  float norm_error = 1;
  float u_zero[m][1] = {0, 0};
  // Iterate estimator until norm_error is small
  while ( norm_error > 1e-5 )
  {
    // Copy over xhat to xhat_old
    float xhat_old[n][1];
    MatrixCopy((float*)xhat, n, 1, (float*)xhat_old);

    // Estimate new state
    stateEstim(u_zero, y, xhat);

    // Calculate error (= xhat - xhat_old)
    float error[n][1];
    MatrixSubtract((float*)xhat, (float*)xhat_old, n, 1, (float*)
       error);

    // Calculate norm of error
    norm_error = VectorNorm((float*)error, n);
  }
}

// Filter method for a moving average filter
float Controller::filter(float y, float Y[max_samples], int order)
{
  // Update vector, Y, of sample histories (zero is current)
  for (uint8_t i = order; i > 0; i = i-1)
  {
    // Shift everything downward
    Y[i] = Y[i-1];
  }
  Y[0] = y;

  // Moving average = (sum of samples) / (# samples)
  float average = 0;
  for (uint8_t i = 0; i <= order; i++)
  {
    average += Y[i];
  }
  average = average/((float)order + 1);

  return average;
}

// Vector norm method
float Controller::VectorNorm(float* Vector, int size)
{
  float squares = 0.0;
  // Sum the squares
  for (uint16_t i = 0; i < size; i++)
  {
    squares += Vector[i]*Vector[i];
  }
```

```
  // norm = sqrt(sum_of_the_squares)
  float norm = sqrt(squares);

  return norm;
}

// Methods from MatrixMath library

// Copy matrix
void Controller::MatrixCopy(float* A, int rowsA, int colsA, float* B
    )
{
        int i, j, k;
        for (i=0;i<rowsA;i++)
                for(j=0;j<colsA;j++)
                {
                        B[colsA*i+j] = A[colsA*i+j];
                }
}

//Matrix Multiplication Routine
// C = A*B
void Controller::MatrixMult(float* A, float* B, int rowsA, int colsA
    , int colsB, float* C)
{
        // C = output matrix = A*B (rowsA x colsB)
        int i, j, k;
        for (i=0;i<rowsA;i++)
                for(j=0;j<colsB;j++)
                {
                        C[colsB*i+j]=0;
                        for (k=0;k<colsA;k++)
                                C[colsB*i+j]= C[colsB*i+j]+A[colsA*i
                                    +k]*B[colsB*k+j];
                }
}

//Matrix Addition Routine
void Controller::MatrixAdd(float* A, float* B, int rowsA, int colsA,
     float* C)
{
        // C = output matrix = A+B
        int i, j;
        for (i=0;i<rowsA;i++)
                for(j=0;j<colsA;j++)
                        C[colsA*i+j]=A[colsA*i+j]+B[colsA*i+j];
}


//Matrix Subtraction Routine
void Controller::MatrixSubtract(float* A, float* B, int rowsA, int
    colsA, float* C)
{
        // C = output matrix = A-B
```

```
  int i, j;
        for (i=0;i<rowsA;i++)
                for(j=0;j<colsA;j++)
                        C[colsA*i+j]=A[colsA*i+j]-B[colsA*i+j];
}


//Matrix Transpose Routine
void Controller::MatrixTranspose(float* A, int rowsA, int colsA,
   float* C)
{
        // C = output matrix = the transpose of A (colsA x rowsA)
        int i, j;
        for (i=0;i<rowsA;i++)
                for(j=0;j<colsA;j++)
                        C[rowsA*j+i]=A[colsA*i+j];
}
```

```
/*
 * HSC.h    - Library for interacting with a
 *            Honeywell HSC Series pressure sensor
 *            Range: -1 to 1 psig
 *            I2C Address: 0x28
 *
 * Usage: HSC pSensor (channel, output)
 *       The argument channel is an integer 1 - 4 to choose the
 *       multiplexer channel the sensor is connected to (only 1 and 2
 *       should be connected). A value of 0 specifies that the sensor
 *       is connected directly (can be used for debugging).
 *
 * Created by Jeff Kornuta, June 8, 2012.
 * Released into the public domain.
 *
 */

#ifndef HSC_H
#define HSC_H

/********************
* Class Declaration *
********************/
class HSC
{
    public:
        HSC(uint8_t sensorChannel);
        double pressure(void);
        uint16_t pressureInt(void);

    protected:
        uint8_t selChan;
};

#endif // HSC_H
```

```
/*
 * HSC.cpp - Library for interacting with a
 *           Honeywell HSC Series pressure sensor
 *           Range: -1 to 1 psig
 *           I2C Address: 0x28
 *
 * Usage: HSC pSensor (channel)
 *      The argument channel is an integer 1 - 4 to choose the
 *      multiplexer channel the sensor is connected to (only 1 and 2
 *      should be connected). A value of 0 specifies that the sensor
 *      is connected directly (can be used for debugging).
 *
 * Created by Jeff Kornuta, June 8, 2012.
 * Released into the public domain.
 *
 */

#include "WProgram.h"
#include "HSC.h"
#include "math.h"
#include "Wire.h"

/*****************************
* Configuration Definitions *
*****************************/
// Sensor address for writing and reading
#define pressAddr 0x28

// I2C multiplexer address
#define mpxAddr 0x70

/***********************
* Function Definitions *
***********************/
HSC::HSC(uint8_t sensorChannel)
{
    // Set protected variable for setting multiplexer channel
    selChan = sensorChannel;
}

// Return pressure from sensor in cmH2O
double HSC::pressure(void)
{
    // Are we really connected through the multiplexer?
    // If so, open up the appropriate multiplexer channel
    if ( selChan != 0 )
    {
        Wire.beginTransmission(mpxAddr);
        Wire.send(selChan);
        uint8_t error = Wire.endTransmission();

        // Let us know if there was a transmission error
        if ( error != 0 )
```

```
        {
            Serial.print("Multiplexer transmission error: ");
            Serial.println(error, DEC);
        }
    }

    // Grab pressure value
    Wire.requestFrom((int) pressAddr, 2);
    uint8_t pressHB = Wire.receive();
    uint8_t pressLB = Wire.receive();

    // Turn bytes into one integer
    uint16_t pressure = (uint16_t) pressHB;
    pressure = ((pressure << 8) | ((uint16_t) pressLB));

    // Convert to psig then cmH2O, after finding output % (see
       datasheet)
    double output = ((double) pressure)/16383.0;
    double psig = 2.5*(output - 0.1) - 1.0;
    double cmH2O = 70.30696*psig;

    return cmH2O;
}

// Return pressure from sensor as an unsigned int
uint16_t HSC::pressureInt(void)
{
    // Are we really connected through the multiplexer?
    // If so, open up the appropriate multiplexer channel
    if ( selChan != 0 )
    {
        Wire.beginTransmission(mpxAddr);
        Wire.send(selChan);
        uint8_t error = Wire.endTransmission();

        // Let us know if there was a transmission error
        if ( error != 0 )
        {
            Serial.print("Multiplexer transmission error: ");
            Serial.println(error, DEC);
        }
    }

    // Grab pressure value
    Wire.requestFrom((int) pressAddr, 2);
    uint8_t pressHB = Wire.receive();
    uint8_t pressLB = Wire.receive();

    // Turn bytes into one integer
    uint16_t pressure = (uint16_t) pressHB;
    pressure = ((pressure << 8) | ((uint16_t) pressLB));

    return pressure;
}
```

```
/*
 * Servo.h - Library for interacting with a
 *           Parker MX80L linear stage
 *
 * Created by Jeff Kornuta, September 18, 2011.
 * Modified by Phillip Johnston, 25 May 2012.
 * Released into the public domain.
 *
*/

#ifndef SERVO_H
#define SERVO_H

#include <stdint.h>

/********************
* Class Declaration *
********************/
class Servo
{
    public:
        Servo(uint8_t quadDecCS, uint8_t dacCS, uint8_t dac);
        void init(void);
        double position(void);
        void move(double volts);
        void zero(void);

    private:
        void _ls7366rConfig(void);
        void _dacConfig(uint8_t power_setting);
        uint8_t _quadDecCS;
        uint8_t _dacCS;
        uint8_t selected_dac;
};

#endif //SERVO_H
```

```
/*
 * Servo.cpp - Library for interacting with a
 *          Parker MX80L linear stage
 *
 * Created by Jeff Kornuta, September 18, 2011.
 * Modified by Phillip Johnston, 22 May 2012
 * Released into the public domain.
 *
*/

#include "WProgram.h"
#include "Servo.h"
#include "LS7366R.h"
#include "HardwareSerial.h"
#include "SPI.h"
```

```
#include "dac5752.h"
#include "math.h"

/**********************
* Object Declarations *
**********************/
DACClass DAC;
LS7366RClass QD;

/****************************
* Configuration Definitions *
****************************/
// configuration vars for the LS7366R
#define MDR0_CONFIG 67   // 0 1 00 00 11
#define MDR1_CONFIG 0    // 00000000


/***********************
* Function Definitions *
***********************/
Servo::Servo(uint8_t quadDecCS, uint8_t dacCS, uint8_t dac)
{
  // Set private vars so (private) methods can initialize properly
  _quadDecCS = quadDecCS;
  _dacCS = dacCS;
  selected_dac = dac;
}

// read position of motor (via LS7366R)
double Servo::position(void)
{
    QD.setupSPI(); //Ensure SPI is configured for this device
    return QD.readPosition(_quadDecCS);
}

// initialize the motor, encoder and DAC chips
void Servo::init(void)
{
    SPI.begin();
    _ls7366rConfig();
    _dacConfig(BIPOLAR_10V);
}

void Servo::_ls7366rConfig(void)
{
    uint8_t MDR0_Val;
    uint8_t MDR1_Val;

    // Set QD CS pin
    QD.setCSPin(_quadDecCS);

    QD.setupSPI(); //Ensure SPI is configured for this device

    QD.setMDR1Reg(MDR1_CONFIG);
```

```
    //delayMicroseconds(200);

    QD.setMDR0Reg(MDR0_CONFIG);
    //delayMicroseconds(200);

    //Now we will get the register vals to make sure everything is
       kosher
    MDR1_Val = QD.getMDR1Reg();
    //delayMicroseconds(200);
    MDR0_Val = QD.getMDR0Reg();

    if (MDR0_CONFIG == MDR0_Val && MDR1_CONFIG == MDR1_Val)
    {
        Serial.print(" > LS7366R Quadrature Decoder (CS pin = ");
        Serial.print( QD.getChipSelectPin(), DEC );
        Serial.println(") successfully configured.");
    }
    else
    {
        Serial.println("LS7366R MDR0 vals:");
        Serial.println(MDR0_CONFIG, BIN);
        Serial.println(MDR0_Val, BIN);
        Serial.println("LS7366R MDR1 vals:");
        Serial.println(MDR1_CONFIG, BIN);
        Serial.println(MDR1_Val, BIN);
        Serial.println("\n*** LS7366R CONFIGURATION FAILURE! ***");
        //while (1);
    }

    // Clear counter value initially
    QD.clear(_quadDecCS);

}

void Servo::_dacConfig(uint8_t power_setting)
{
    uint8_t newly_powered_DAC;

    //Initialize SPI for DAC
    DAC.setCSPin(_dacCS);

    DAC.setupSPI(); //Ensure SPI is configured for this device
    DAC.enableSDO();

    //See if another DAC is powered on so we don't overwrite
       settings
    //Note:  For now, it will make sure the DACs use the same range.
    //If this is not desired, the code needs to be reworked.

    DAC.setOutputRange((uint8_t) selected_dac, (uint32_t)
       power_setting);

    uint32_t configuredDACPower = DAC.getPowerControl();
```

```
    uint8_t powered_DAC = configuredDACPower & 0xF; //Get the
       relevant portion

    if(selected_dac == DAC_A)
    {
      DAC.setPowerControl(POWER_DAC_A | powered_DAC); //Power up DAC
          A
      newly_powered_DAC = POWER_DAC_A;
    }
    else if(selected_dac == DAC_B)
    {
      DAC.setPowerControl(POWER_DAC_B | powered_DAC);
      newly_powered_DAC = POWER_DAC_B;
    }
    else if(selected_dac == DAC_ALL)
    {
      DAC.setPowerControl(POWER_DAC_ALL);
      newly_powered_DAC = POWER_DAC_ALL;
    }

    //Check the value of the power control register.
    uint32_t dacCheckPower = DAC.getPowerControl();

    if( (dacCheckPower & 0xF) == (newly_powered_DAC | powered_DAC))
    {
      Serial.print(" > DAC power control (DAC ");
      Serial.print( selected_dac, DEC );
      Serial.println(") successfully configured.\n");
    }
    else
    {
        Serial.println("DAC5752 Power Control Reg Values:");
        Serial.print("Config Value: ");
        Serial.print((selected_dac | powered_DAC), BIN);
        Serial.print("\nRegister Value: ");
        Serial.print(dacCheckPower & 0xF, BIN);
        Serial.println("\n\n*** DAC5752 CONFIGURATION FAILUE! ***");
        //while(1) {} //Wait indefinitely in error loop
    }

    DAC.disableSDO();
}

// apply +-10 V (double) signal to motor
void Servo::move(double volts)
{
    // Code for UNIPOLAR_10V
    //float scale = volts / 10.0;
    //int16_t value = (int16_t) floor(scale * 0xFFFF); //Scale *
       value for 10V.

    // Code for BIPOLAR_10V
    int16_t value;
```

```
    if ( volts >= 0 )
    {
      // Set saturation
      if ( volts > 10.0 ) { volts = 10.0; }

      // Scale value for 10 V
      value = (int16_t) ( volts / 10.0 * 0x7FFF );
    }
    else
    {
      // Set saturation
      if ( volts < -10.0 ) { volts = -10.0; }

      // Scale value for 10 V, then take 2's complement
      value = (int16_t) ( abs(volts) / 10.0 * 0x8000 );
      value = ( ~value ) + 1;
    }

    // Set DAC output
    DAC.setValue(selected_dac, value);
    //DAC.disableSDO}
}

// Zero-out (clear) QD counter value
void Servo::zero(void)
{
  QD.setupSPI();
  QD.clear(_quadDecCS);
}
```

```
/*
* DAC 5752 Library
* Written by Phillip Johnston, modified/fixed by Jeff Kornuta
* 3 March 2012
*
* dac5752.h
*
*/
#ifndef _DAC_5752_H
#define _DAC_5752_H

#include <stdint.h>
#include "SPIDevice.h"

/*******************
* Type Definitions *
*******************/
typedef enum outputRanges
{
    UNIPOLAR_5V = 0,
    UNIPOLAR_10V,
    UNIPOLAR_10p8V,   // +10.8 V
    BIPOLAR_5V,
    BIPOLAR_10V,
```

```
    BIPOLAR_10p8V     // +-10.8 V
} dac_output_range_t;

/*********************
* Define Statements *
*********************/
//Debugging Definitions
//#define DAC_DEBUG

//Register Definitions
#define RW 128 //BIT 7*, Bit 6 is "zero"
#define REG2 32 //Bit 5
#define REG1 16 //Bit 4
#define REG0 8 //Bit 3
#define __A2 (1 << 2) //Bit 2
#define __A1 (1 << 1) //Bit 1
#define __A0 (1 << 0) //Bit 0
        /* NOTE:
        A2 .. A0 were renamed with two "__" in front, because
           leaving them as "A2",
        "A1", and "A0" caused them to be valued at 16, 15, and 14
           respectively.
        This causes communication with the control register and DAC
           B to fail.*/

// Power Up Definitions
#define POWER_DAC_A 1
#define POWER_DAC_B (1 << 2)
#define POWER_DAC_ALL POWER_DAC_A | POWER_DAC_B

//Channel Definitions
#define DAC_A 0
#define DAC_B __A1
#define DAC_ALL __A2

//Control Definitions
#define OUTPUT_RANGE_SEL REG0
#define CONTROL (REG0 | REG1)
#define CONTROL_SET (CONTROL | __A0)
#define POWER_CONTROL REG1

#define CTRL_SDO_DISABLE 1
#define CTRL_CLAMP_EN 4
#define CTRL_THERMAL_SHUTDOWN 8

/*********************
* Class Declaration *
*********************/
class DACClass : public SPIDevice
{
    public:
        DACClass();
        ~DACClass();
```

```
        void setupSPI();
        void setOutputRange(uint8_t address, uint8_t voltage_range);
        uint32_t getOutputRange(uint8_t address);
        void setControl();
        void setPowerControl(uint8_t channels);
        void setValue(uint8_t address, uint16_t value);
        uint32_t getPowerControl();
        uint32_t getControl();
        void powerDownDAC(uint8_t channels);
        void enableSDO();
        void disableSDO();
};

#endif
```

```
/*
* DAC 5752 Library
* Written by Phillip Johnston, modified/fixed by Jeff Kornuta
* 3 March 2012
*
* dac5752.cpp
*
*/

#include "dac5752.h"
#include "SPI.h"

/*********************
* Define Statements *
*********************/
#ifdef DAC_DEBUG
    #define debugf(msg) Serial.print msg
#else
    #define debugf(msg)
#endif

/************************
* Function Definitions *
************************/
DACClass::DACClass()
{
    debugf(("Initializing DAC object...\n"));
}

DACClass::~DACClass()
{
    //SPI.end(); //Release SPI stuff
}

void DACClass::setupSPI()
{
    //SPI Settings for DAC5752
    SPI.setBitOrder(MSBFIRST);
    SPI.setDataMode(SPI_MODE2);
```

```
    SPI.setClockDivider(SPI_CLOCK_DIV4);
}

uint32_t DACClass::getPowerControl()
{
    _send(RW | POWER_CONTROL, 0, 0);
    return getLastTransmissionResult();
}

void DACClass::setPowerControl(uint8_t channels)
{
    _send(POWER_CONTROL, 0, channels & 15);
     delayMicroseconds(10); //Datasheet specifies a delay of 10
        microseconds befoer issuing another command.
}

uint32_t DACClass::getControl()
{
     _send( RW | CONTROL_SET, 0, 0 );
     return getLastTransmissionResult();
}

void DACClass::setControl()
{
    _send(CONTROL_SET,0, CTRL_THERMAL_SHUTDOWN | CTRL_CLAMP_EN);
}

void DACClass::setValue(uint8_t address, uint16_t val)
{
    _send(address,(uint8_t)(val >> 8) & 0xFF,(uint8_t)(val) & 0xFF);
}

void DACClass::setOutputRange(uint8_t address, uint8_t voltage_range
   )
{
    _send(OUTPUT_RANGE_SEL | address, 0, voltage_range);
}

void DACClass::disableSDO()
{
  _send(CONTROL_SET, 0, CTRL_THERMAL_SHUTDOWN | CTRL_CLAMP_EN |
     CTRL_SDO_DISABLE);
}

void DACClass::enableSDO()
{
  setControl();
}
```

```
/*
* LS7366R Class
* Initial functionality created by Jeff Kornuta
* Packaged into class and expanded by Phillip Johnston
*
```

```
* 25 May 2012
*
* LS7366R.h
*/

#ifndef _LS7366R_H
#define _LS7366R_H

#include <stdint.h>
#include "SPIDevice.h"

/********************
* Define Statements *
********************/
// LS7366R OP-Codes
#define CLEAR_COUNTER 32 //8
#define CLEAR_STATUS 48
#define READ_COUNTER 96
#define READ_STATUS 112 //122
#define READ_MDR0 72
#define READ_MDR1 80
#define WRITE_MDR0 136
#define WRITE_MDR1 144

/********************
* Class Declaration *
********************/
class LS7366RClass : public SPIDevice
{
    public:
        LS7366RClass();
        ~LS7366RClass();
        void clear(uint8_t cs_pin);
        void setupSPI();
        double readPosition(uint8_t cs_pin);
        void setMDR0Reg(uint8_t config_val);
        void setMDR1Reg(uint8_t config_val);
        uint8_t getMDR0Reg();
        uint8_t getMDR1Reg();
};
#endif
```

```
/*
* LS7366R Class
* Code written by Jeff Kornuta
* Packaged and expanded by Phillip Johnston
*
* 25 May 2012
*
* LS7366R.cpp
*/

#include "LS7366R.h"
#include "SPI.h"
```

```
#include "SPIDevice.h"

/*********************
* Define Statements *
*********************/
#ifdef LS7366R_DEBUG
    #define debugf(msg) Serial.print msg
#else
    #define debugf(msg)
#endif

/************************
* Function Definitions *
************************/
LS7366RClass::LS7366RClass()
{
    debugf(("Initializing LS7366R object...\n"));
}

LS7366RClass::~LS7366RClass()
{
    //Not destructing anything currently
}

void LS7366RClass::clear(uint8_t cs_pin)
{
  // Clear counter
  CSpin = cs_pin;
  _send(CLEAR_COUNTER, 0);
}

void LS7366RClass::setupSPI()
{
   SPI.setBitOrder(MSBFIRST);
   SPI.setDataMode(SPI_MODE1);
   SPI.setClockDivider(SPI_CLOCK_DIV16);
}

void LS7366RClass::setMDR0Reg(uint8_t config_val)
{
    _send(WRITE_MDR0, config_val, 0);
}

void LS7366RClass::setMDR1Reg(uint8_t config_val)
{
    _send(WRITE_MDR1, config_val, 0);
}

double LS7366RClass::readPosition(uint8_t cs_pin)
{
    int32_t count;

    // Know which QD to use!
    CSpin = cs_pin;
```

```
    // send read counter command
    count = _transfer( READ_COUNTER );

    // return value in mm (change sign for correct direction)
    return (-1.0*(double)count / 2000.0); //0.5um / count
}

uint8_t LS7366RClass::getMDR0Reg()
{
    uint32_t return_data = _transfer(READ_MDR0, 0, 0);
    return (uint8_t) (return_data & 0xFF);
}

uint8_t LS7366RClass::getMDR1Reg()
{
    uint32_t return_data = _transfer(READ_MDR1, 0, 0);
    return (uint8_t) (return_data & 0xFF); //double check this to
       make sure we're grabbing the right byte
}
```

```
/*
* SPI Device Class
* Created by Phillip Johnston
* 31 May 2012
*
* This class was created to allow common function calls
* to be encapsulated in a base SPIDevice Class
*
* SPIDevice.h
*
*/

#ifndef SPI_DEVICE_H
#define SPI_DEVICE_H

#include <stdint.h>

/*********************
* Define Statements *
*********************/
//#define SPIDEV_DEBUG

/*********************
* Class Declaration *
*********************/
class SPIDevice
{
public:
    SPIDevice();
    ~SPIDevice();
    void setCSPin(uint8_t cs_pin);
    //virtual void setupSPI();
    uint8_t getChipSelectPin();
```

```
    uint32_t getLastTransmissionResult(void);

protected:
    uint8_t CSpin;
    inline void _enableChipSelect();
    inline void _disableChipSelect();
    void _send(uint8_t a, uint8_t b, uint8_t c);
    void _send(uint8_t a, uint8_t b);
    uint32_t _transfer(uint8_t a, uint8_t b, uint8_t c);
    int32_t _transfer(uint8_t cmd);
};

#endif //SPI_DEVICE_H
```

```
/*
* SPI Device Class
* Created by Phillip Johnston
* 31 May 2012
*
* This class was created to allow common function calls
* to be encapsulated in a base SPIDevice Class
*
* SPIDevice.cpp
*
*/

#include "SPIDevice.h"
#include "SPI.h"

/*********************
* Define Statements *
*********************/
//#define SPIDEV_DEBUG

#ifdef SPIDEV_DEBUG
    #define debugf(msg) Serial.print msg
#else
    #define debugf(msg)
#endif

/************************
* Function Definitions *
************************/
SPIDevice::SPIDevice()
{
    //Nothing to construct
}

SPIDevice::~SPIDevice()
{
    //Nothing to destruct
}

/*
```

```
void SPIDevice::setupSPI()
{
    //We're gonna default to SPI_MODE1 for now.
    //TODO:  Make this more robust/generic
    SPI.setBitOrder(MSBFIRST);
    SPI.setDataMode(SPI_MODE1);
    SPI.setClockDivider(SPI_CLOCK_DIV16);
}
*/

void SPIDevice::setCSPin(uint8_t cs_pin)
{
    CSpin = cs_pin;
    pinMode(CSpin, OUTPUT);
    _disableChipSelect();
}

uint32_t SPIDevice::getLastTransmissionResult()
{
    return _transfer(0x18, 0, 0);
}

inline void SPIDevice::_enableChipSelect()
{
    debugf(("Enable Chip Select Function enetered.\n"));
    digitalWrite(CSpin, LOW);
}

inline void SPIDevice::_disableChipSelect()
{
    debugf(("Disable Chip Select Function enetered.\n"));
    digitalWrite(CSpin, HIGH);
}

uint32_t SPIDevice::_transfer(uint8_t a, uint8_t b, uint8_t c)
{
    debugf(("Sending the following message over SPI:"));
    debugf(("\nMSB:  "));
    debugf((a, HEX));
    debugf(("\nMID:  "));
    debugf((b, HEX));
    debugf(("\nLSB:  "));
    debugf((c, HEX));
    debugf(("\n"));

    _enableChipSelect();

    uint8_t a_ret = SPI.transfer(a);
    uint8_t b_ret = SPI.transfer(b);
    uint8_t c_ret = SPI.transfer(c);

    _disableChipSelect();

    //Store the returned values as a single uint32_t
```

```
    uint32_t ret_val = (uint32_t) a_ret;
    ret_val = ((ret_val << 8) | ((uint32_t) b_ret));
    ret_val = ((ret_val << 8) | ((uint32_t) c_ret));

    return ret_val;
}

// Function specifically to read position of QD
int32_t SPIDevice::_transfer(uint8_t cmd)
{
  uint8_t result[4] = {0, 0, 0, 0};

  // Send cmd
  _enableChipSelect();
  SPI.transfer(cmd);

  // Receive bytes back
  result[0] = SPI.transfer(0x00);
  result[1] = SPI.transfer(0x00);
  result[2] = SPI.transfer(0x00);
  result[3] = SPI.transfer(0x00);
  _disableChipSelect();

  // Debug
  /*
  Serial.print(result[0], HEX);
  Serial.print(" ");
  Serial.print(result[1], HEX);
  Serial.print(" ");
  Serial.print(result[2], HEX);
  Serial.print(" ");
  Serial.print(result[3], HEX);
  Serial.println(" ");
  */

  // Store result as single 32-bit (unsigned) int
  int32_t counter = result[0];
  counter = ((counter << 8) | result[1] );
  counter = ((counter << 8) | result[2] );
  counter = ((counter << 8) | result[3] );

  // Return value
  return counter;
}

void SPIDevice::_send(uint8_t a, uint8_t b, uint8_t c)
{
    debugf(("Sending the following message over SPI:"));
    debugf(("\nMSB:  "));
    debugf((a, HEX));
    debugf(("\nMID:  "));
    debugf((b, HEX));
    debugf(("\nLSB:  "));
    debugf((c, HEX));
```

```
    debugf(("\n"));

    _enableChipSelect();

    SPI.transfer(a);
    SPI.transfer(b);
    SPI.transfer(c);

    _disableChipSelect();
}

void SPIDevice::_send(uint8_t a, uint8_t b)
{
    debugf(("Sending the following message over SPI:"));
    debugf(("\nMSB:  "));
    debugf((a, HEX));
    debugf(("\nLSB:  "));
    debugf((b, HEX));

    _enableChipSelect();

    SPI.transfer(a);
    SPI.transfer(b);

    _disableChipSelect();
}

uint8_t SPIDevice::getChipSelectPin()
{
    return CSpin;
}
```

B.5 Controller Design Software

```
%
%   mpcMimo.m
%
%   GENERATE MPC PARAMETERS FROM MODEL
%       JK 3/7/13
%
%   REQUIRED:
%       - FUNCTION 'printCMatrix.m' FOR MATRIX PRINTING
%       - IDDATA OBJECT 'mpd' WITH SYSTEM MODEL
%

clear all;

% load model data into SS object
[G,H,C,D] = ssdata(mpd); % = ssdata(mp);
Ts = 3.333333333e-3; % sampling time in sec

n = size(G,1); % number of states
```

```
m = size(H,2); % number of inputs
p = size(C,1); % number of outputs

sysd = ss(G,H,C,D,Ts);
sysdtot = ss(G,H,eye(n),zeros(n,m),Ts);

% MPC parameters
Hp = 5;   % prediction horizon

% Q and R weighting matrices defined as lines along diagonal
%   specify line parameters
qi1=0.5; qf1=0.6; m1=(qf1-qi1)/Hp; % initial/final Q value, slope on
    error 1
qi2=0.8; qf2=0.7; m2=(qf2-qi2)/Hp; % initial/final Q value, slope on
    error 2
ri1=1; rf1=1; mr1=(rf1-ri1)/Hp; % initial/final R value, slope on
   input 1
ri2=1; rf2=1; mr2=(rf2-ri2)/Hp; % initial/final R value, slope on
   input 2

% initialize Q and R, create diagonal row, then make matrix
Q = [qi1 qi2]; % output error penalty
R = [ri1 ri2]; % control input penalty
for i=2:Hp,
    Q = [Q [qi1+m1*i qi2+m2*i]];
    R = [R [ri1+mr1*i ri2+mr2*i]];
end;
Q = diag(Q);
R = diag(R);

% calculate Kca
Kca = C*G;
for i = 2:Hp,
    Kca = [Kca; C*G^i];
end;

% calculate Kcab
dim = size(C*H);
for i = 1:Hp, % for each row
    for j = 1:Hp, % for each column
        % build first column element of row
        if j == 1,
            temp = C*G^(i-j)*H;
            continue;
        end;
        % fill in rest of columns
        if i >= j,
            temp = [temp C*G^(i-j)*H];
        else
            temp = [temp zeros(dim)];
        end;
    end;
    % add row
    if i == 1,
```

```
        Kcab = temp;
    else
        Kcab = [Kcab; temp];
    end;
end;

% calculate K
A = diag(1*ones(1,p*Hp)) + diag(-1*ones(1,p*Hp-1),-1); % for DeltaU
    instead of U
K = inv(Kcab'*Q*Kcab+R)*Kcab'*Q;
K1 = K(1:m,:); % take m rows

%
% create kalman estimator gain, L
Qn = 1e-2*eye(2);
% measurement covariance matrix, Rn, taken from experimental data
Rn = 1e-3*diag([.1413 0.1413]);
[kest,L,P] = kalman(sysd,Qn,Rn);

% make new combined system
Gbar = [G -H*K1*Kca; L*C G-H*K1*Kca-L*C];
Hbar = [H*K1; H*K1];
Cbar = [C zeros(p,n)];
sysdbar = ss(Gbar,Hbar,Cbar,zeros(p,Hp*p),Ts);

%
% test output to some input
tspan = 0:Ts:5;
N = length(tspan);

yd1 = 1*sin(2*pi*0.8*tspan) - cos(2*pi*0.7*tspan) + 2;
yd2 = 1*sin(2*pi*0.6*tspan) - cos(2*pi*0.5*tspan) + 5;

% make additional "ghost" values at end for prediction
yd = [yd1 yd1(N)*ones(1,Hp);
      yd2 yd2(N)*ones(1,Hp)];

% build input vector, column by column
for j = 1:N, % for each column
    for i = 1:Hp, % for each row
        if i == 1,
            temp = yd(:,j+i);
        else
            temp = [temp; yd(:,j+i)];
        end;
    end;
    % add column
    if j == 1,
        Yd = temp;
    else
        Yd = [Yd temp];
    end;
end;
%
```

```
[Y,T,X] = lsim(sysdbar,Yd,tspan);

% variable for simulink simulation
ts = timeseries(Yd,tspan);
% create separate time series with just yd, not Yd
tsyd = timeseries(Yd(1:p,:),tspan);

% % plot results
stairs(T,[yd1' yd2' Y]);
legend('yd1','yd2','y1','y2');

% Oputput relative matrices
fprintf(1, '\n\n*** Matrices in C form: ***\n\n');
fprintf(1, ['float G[n][n] =' printCMatrix(G) '\n']);
fprintf(1, ['float H[n][m] =' printCMatrix(H) '\n']);
fprintf(1, ['float C[p][n] =' printCMatrix(C) '\n']);
fprintf(1, ['float L[n][p] =' printCMatrix(L) '\n']);
fprintf(1, '\n');
fprintf(1, ['float Kca[Hp*p][n] =' printCMatrix(Kca) '\n']);
fprintf(1, ['float K1[m][Hp*p] =' printCMatrix(K1) '\n\n']);
```

```
% printCMatrix.m

function [ strOutput ] = printCMatrix( A )
% Print matrix A in a C way

% Get matrix dims from A
[m,n] = size(A);

strOutput = ' { ';

% Print the matrix
%fprintf(1,'{ ');
for i = 1:m,
    if i ~= 1,
        strOutput = [strOutput '\t'];
    end;
    for j = 1:n,
        if i == m && j == n,
            %fprintf(1,'%.4f }\n', A(i,j));
            strOutput = [strOutput sprintf('%.4f };\n', A(i,j))];
        else
            %fprintf(1,'%.4f, ', A(i,j));
            strOutput = [strOutput sprintf('%.4f, ', A(i,j))];
        end;
    end;
    if i ~= m,
        %fprintf(1,'},\n');
        strOutput = [strOutput '\n'];
    end;
end
```

APPENDIX C

ANCILLARY MATERIAL FOR CHAPTER IV

C.1 Frequency-Following Conditions with L-NAME

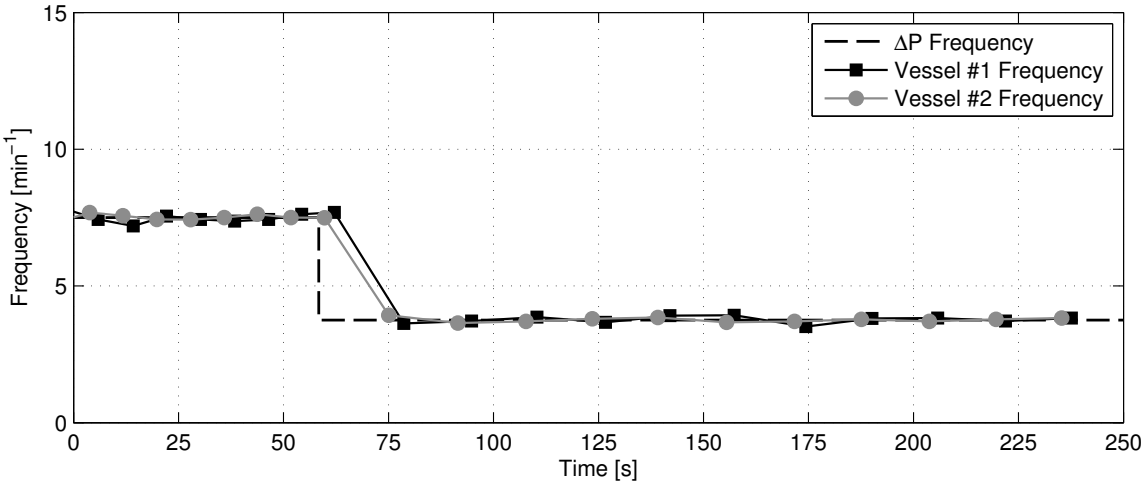


Figure C.1: Point-to-point frequency data over time for two isolated vessel preparations transitioning from 0.125 Hz (7.5 min^{-1}) to 0.0625 Hz (3.75 min^{-1}) after being exposed externally to L-NAME (10^{-4}), suggesting nitric oxide synthase has no effect on the syncing of phase contractions.

C.2 Combined Ramp and Sine Data for All Conditions

In addition to the 3-min ramps performed in Chapter 4 to determine shear stress sensitivity, while in Temple, TX the author used the ELPS to impose pressure gradient ramps from 0 to 3 cmH_2O over 30 sec and 90 sec while simultaneously holding the average transmural pressure constant at both 3 and 5 cmH_2O on 5 vessels in 2013. Each ramp was preceded by 5 min of a zero axial pressure gradient at that respective average transmural pressure to allow the vessel to equilibrate again and resume contraction. In addition, the flow rate through the vessel (as well as the shear stress imposed on the vessel) was estimated *post hoc*

in 5-sec time windows throughout the condition. In this way, by applying varying slopes of transaxial pressure gradient, the author could test whether or not the rate of applied shear stress had a noticeable effect on contractile function. Raw data plots of the ramp conditions may be seen in Appendix C.3, including two 3-min ramp experiments performed in 2014 in the LLBB.

Likewise, in addition to the sine wave conditions described in Chapter 4, four other conditions were tested on the 5 vessels following the shear sensitivity ramps in 2013: $f =$ 0.25 Hz (or 15 min^{-1}, which is within the observed range of lymphatics [11, 30]) for 5 min, with both $A = 0.5$ and $A = 1$ cmH_2O, $C = 1$ cmH_2O, and a constant average transmural pressure, P_{avg} = 3 cmH_2O. For each condition, the mean flow rate through the vessel was calculated, as well as wall shear stress value over that time. Besides testing an additional frequency, the author wanted to see if increasing the peak amplitudes of the sine waves also had an effect on contractile function. Raw data plots of the sine conditions may be seen in Appendix C.4, including two frequency conditions performed recently in 2014 in the LLBB.

Ultimately, the combined ramp data did not display any significant changes in the typical isolated functional parameters (normalized diastolic diameter, normalized frequency, normalized amplitude, tone, normalized systolic diameter, and fractional pump flow) with respect to different ramp slopes [Fig. C.2]. Although there are several instances where the 30-sec ramp condition displayed a slight increase (*e.g.* normalized frequency, tone, and fractional pump flow) and the mean shear stress was highest, these results remain inconclusive. Because this condition contained the fewest amount of calculated shear stress points and the fewest amount of contractions due to the short time frame, the data observed may not be as reliable as the other conditions. It is partially for this reason that the 3-min ramp condition was used for the shear sensitivity calculations. Additionally, since nitric oxide (NO) has been known to be a primary regulator of lymphatic function [20], it could be that not enough of this vasodilator had been released during this condition to affect contractile

activity. Further investigation on this topic may be warranted, albeit with a more clever experimental design

As for the combined sine wave data, very little difference was seen between the sine waves with $A = 0.5$ and $A = 1$ cmH_2O [Fig. C.3]. Additionally, because the $A = 1$ cmH_2O conditions occurred later on in the experiment, they were more likely to develop trapped air bubbles, reducing the effective number of samples that could be used. Also, the mean shear stress that occurred during these conditions was also low compared to the steady flow control, reducing the amount of conclusions that could be made about the data. Likewise, the $A = 0.5$ cmH_2O, $f = 0.25$ Hz condition also displayed a mean wall shear stress less-comparable to the steady flow control; thus, the analysis in Chapter 4 focused primarily on the $f = 0.0625$ and $f = 0.125$ Hz conditions. In short, the author believes future experiments with sine waves should focus on those with larger amplitude (like those in Fig. 4.5) in order to more clearly study the mechanism driving contractile coordination and suppressing the inhibition of contraction frequency.

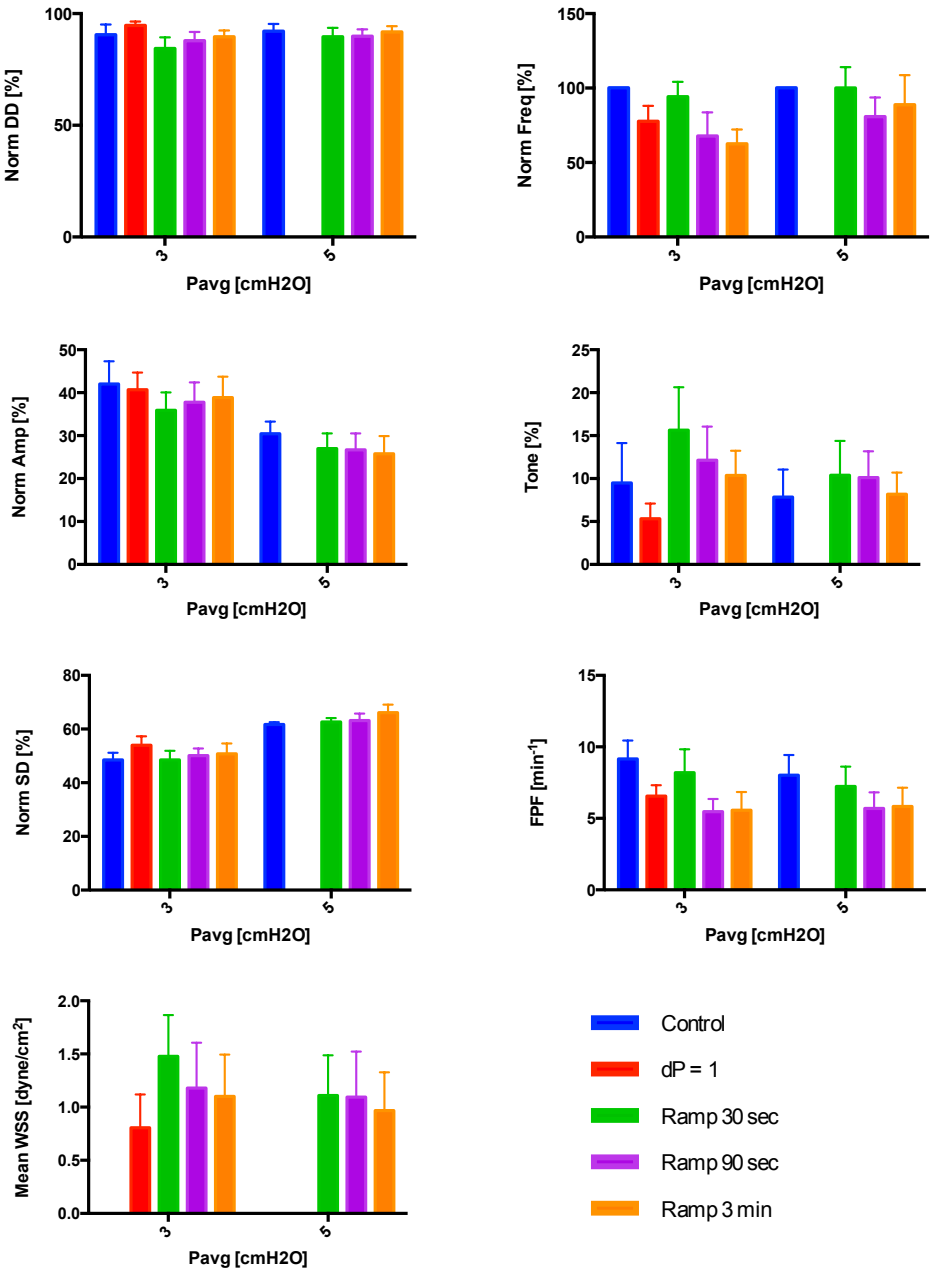


Figure C.2: Combined data from all ramp experiments in Temple, TX.

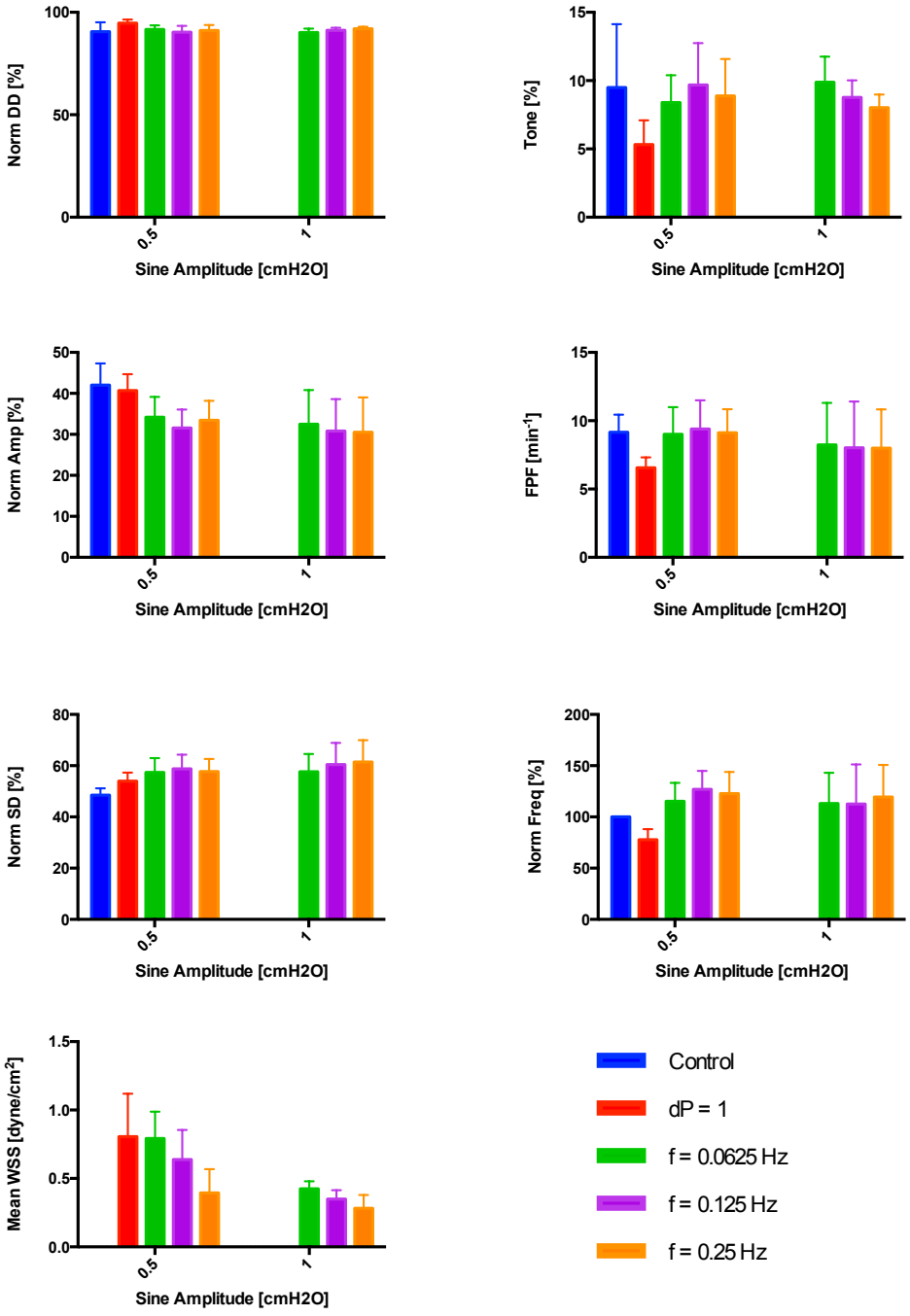


Figure C.3: Combined data from all sine wave experiments in Temple, TX.

C.3 Ramp Data for Individual Vessels

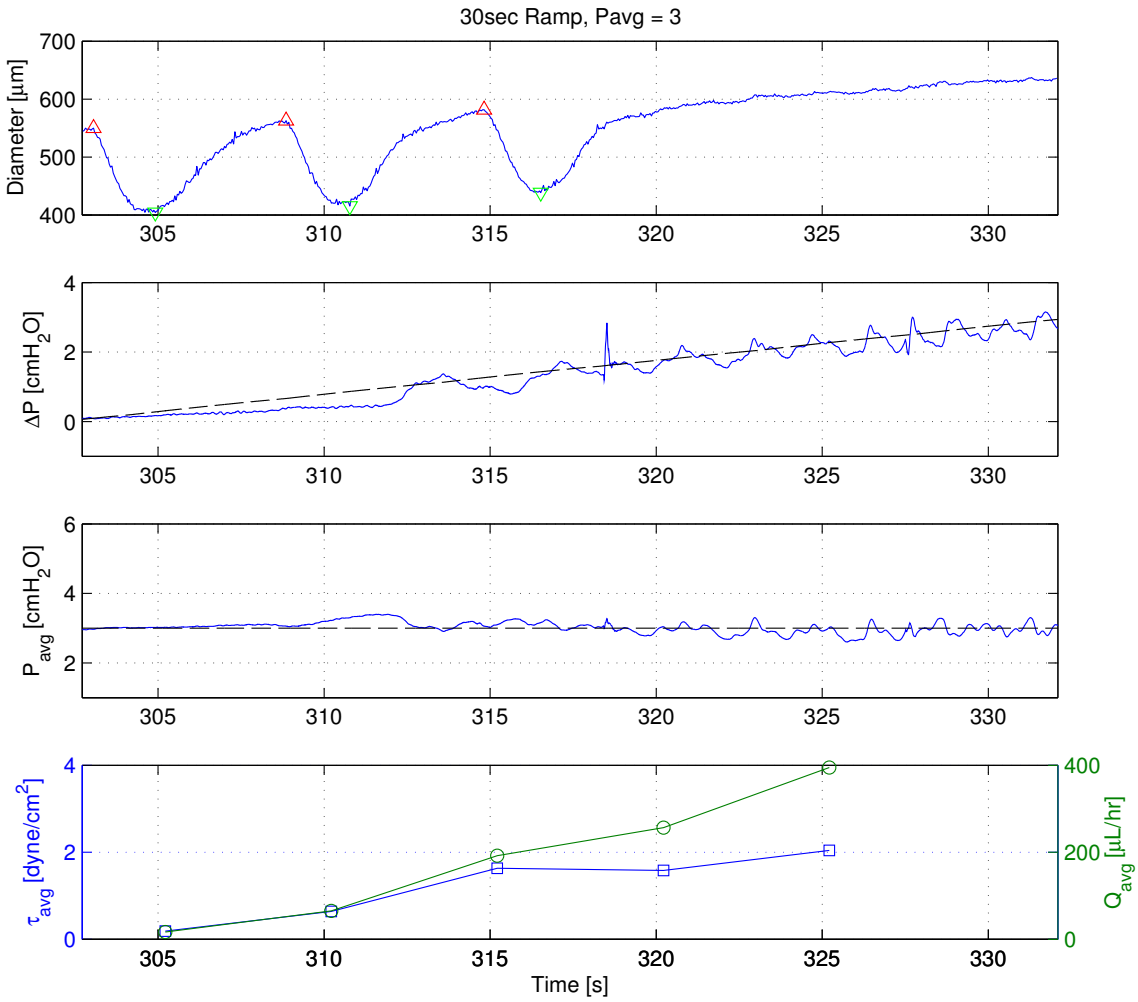


Figure C.4: 2013-08-27: 30-sec Ramp, $P_{\text{avg}} = 3$ cmH_2O.

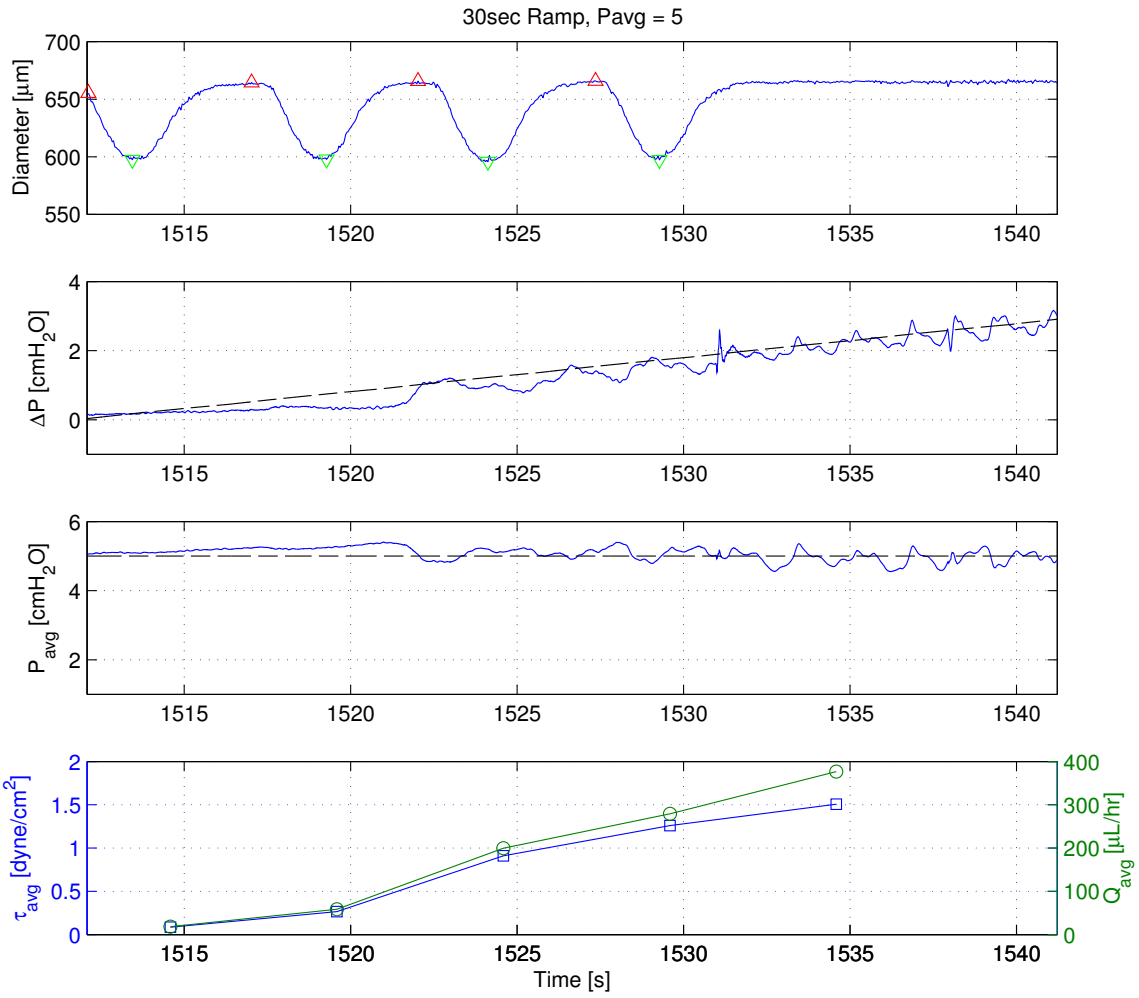


Figure C.5: 2013-08-27: 30-sec Ramp, $P_{\text{avg}} = 5$ cmH_2O.

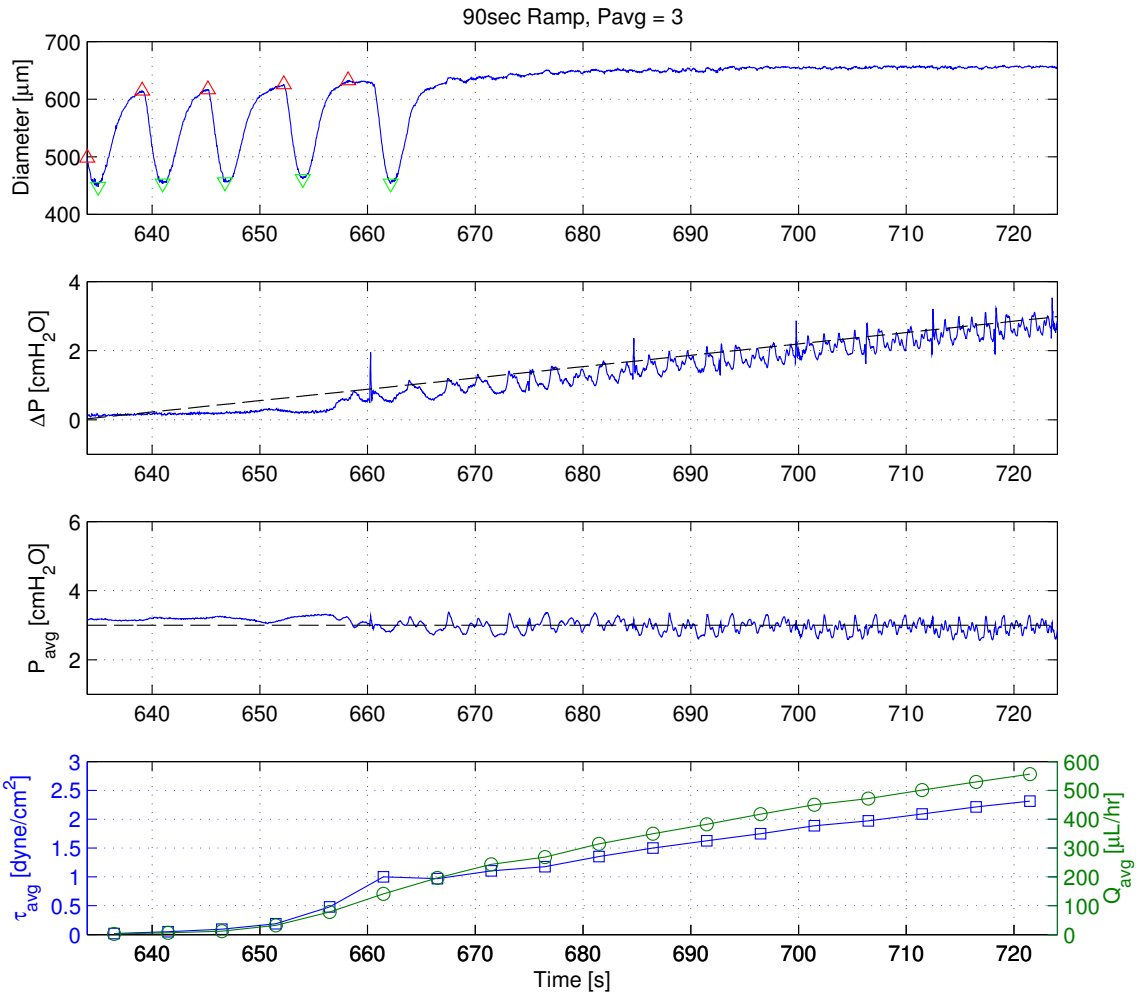


Figure C.6: 2013-08-27: 90-sec Ramp, $P_{\text{avg}} = 3$ cmH_2O.

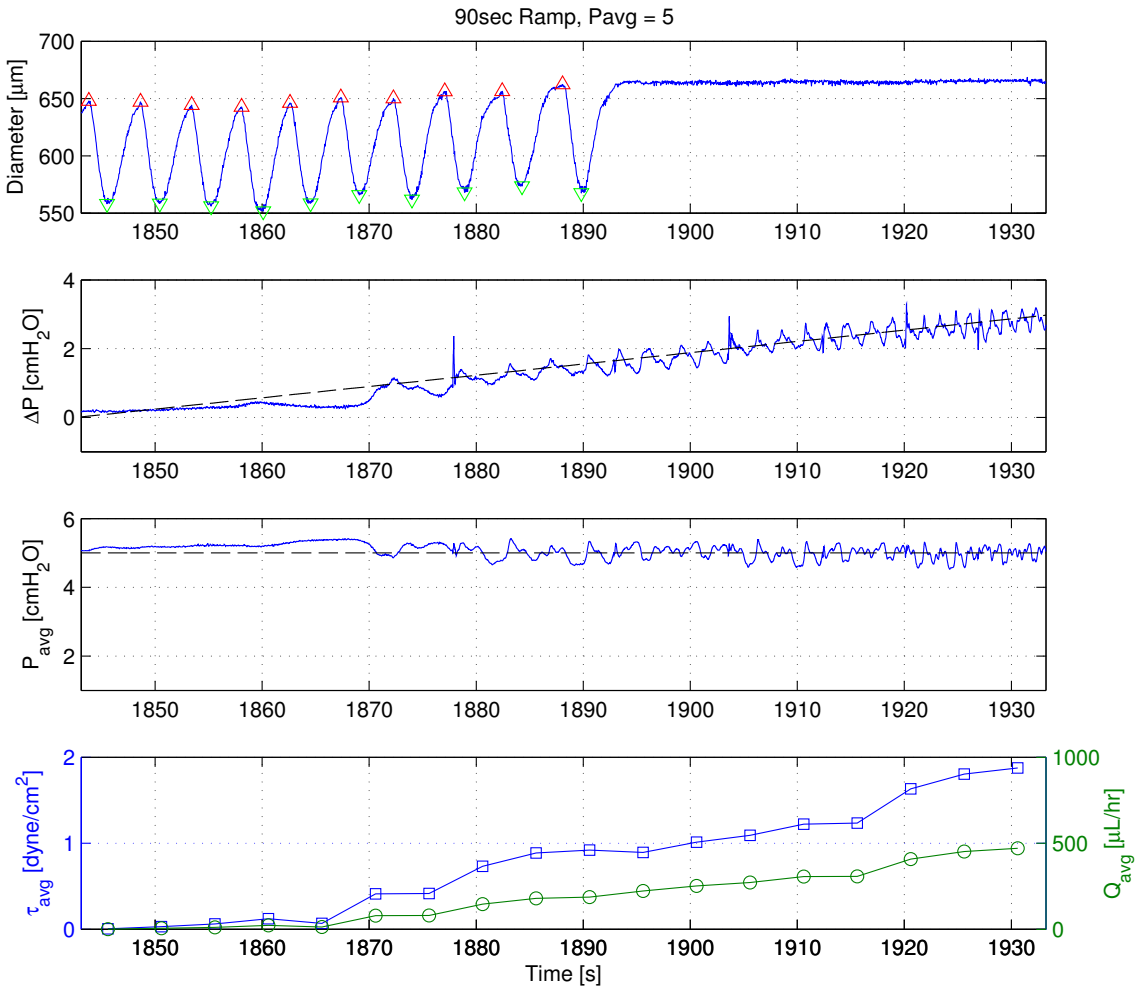


Figure C.7: 2013-08-27: 90-sec Ramp, $P_{\text{avg}} = 5$ cmH_2O.

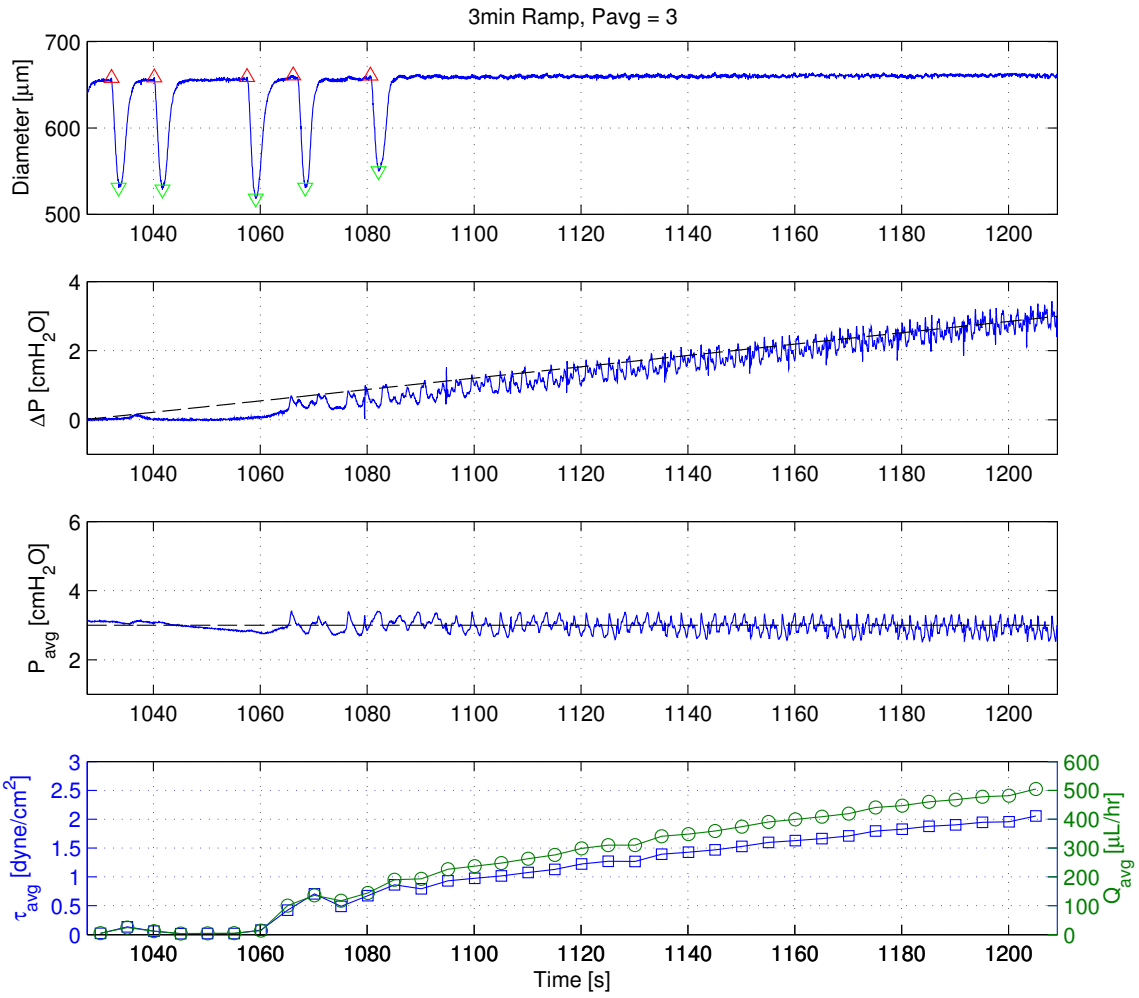


Figure C.8: 2013-08-27: 3-min Ramp, $P_{\text{avg}} = 3$ cmH_2O.

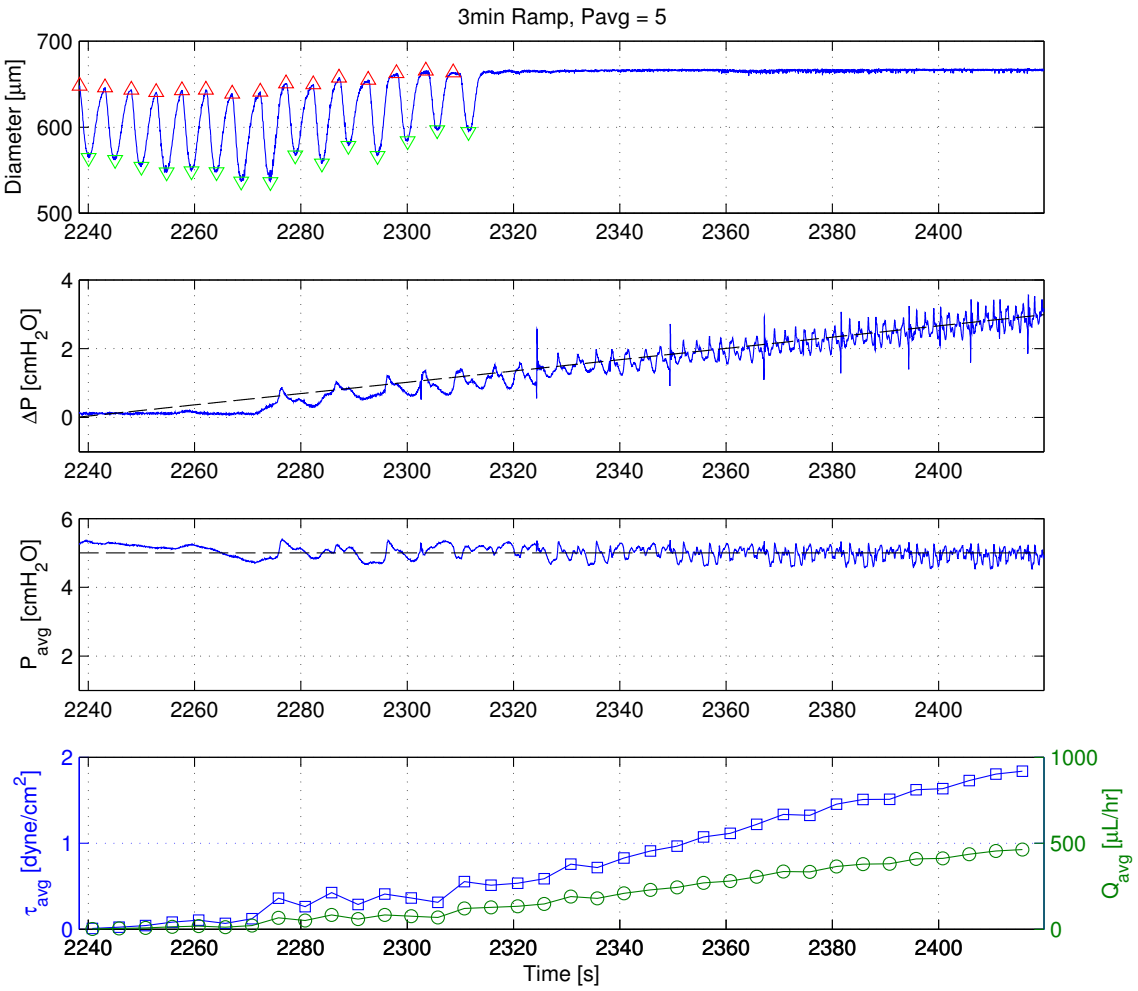


Figure C.9: 2013-08-27: 3-min Ramp, $P_{\text{avg}} = 5$ cmH_2O.

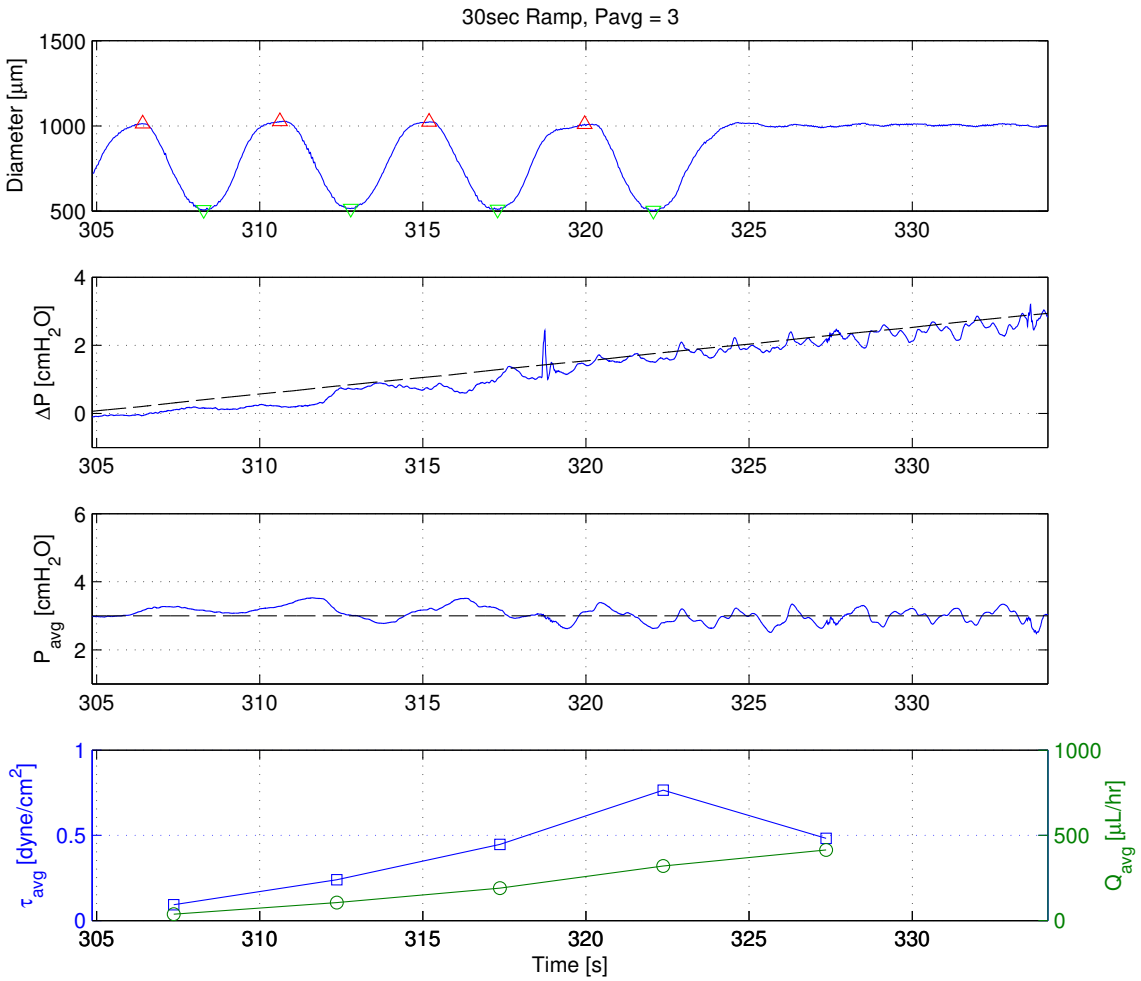


Figure C.10: 2013-08-28: 30-sec Ramp, $P_{\text{avg}} = 3$ cmH_2O.

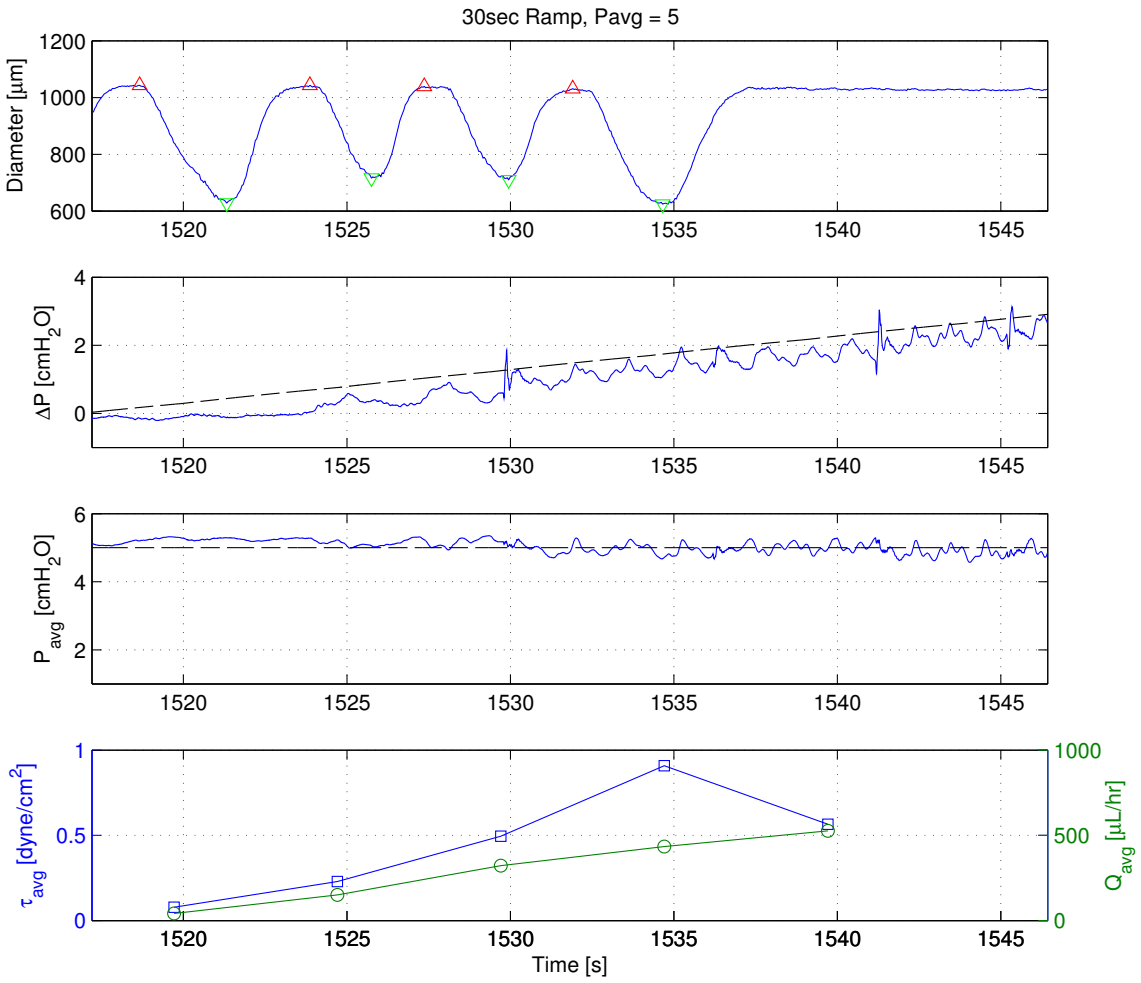


Figure C.11: 2013-08-28: 30-sec Ramp, $P_{\text{avg}} = 5$ cmH_2O.

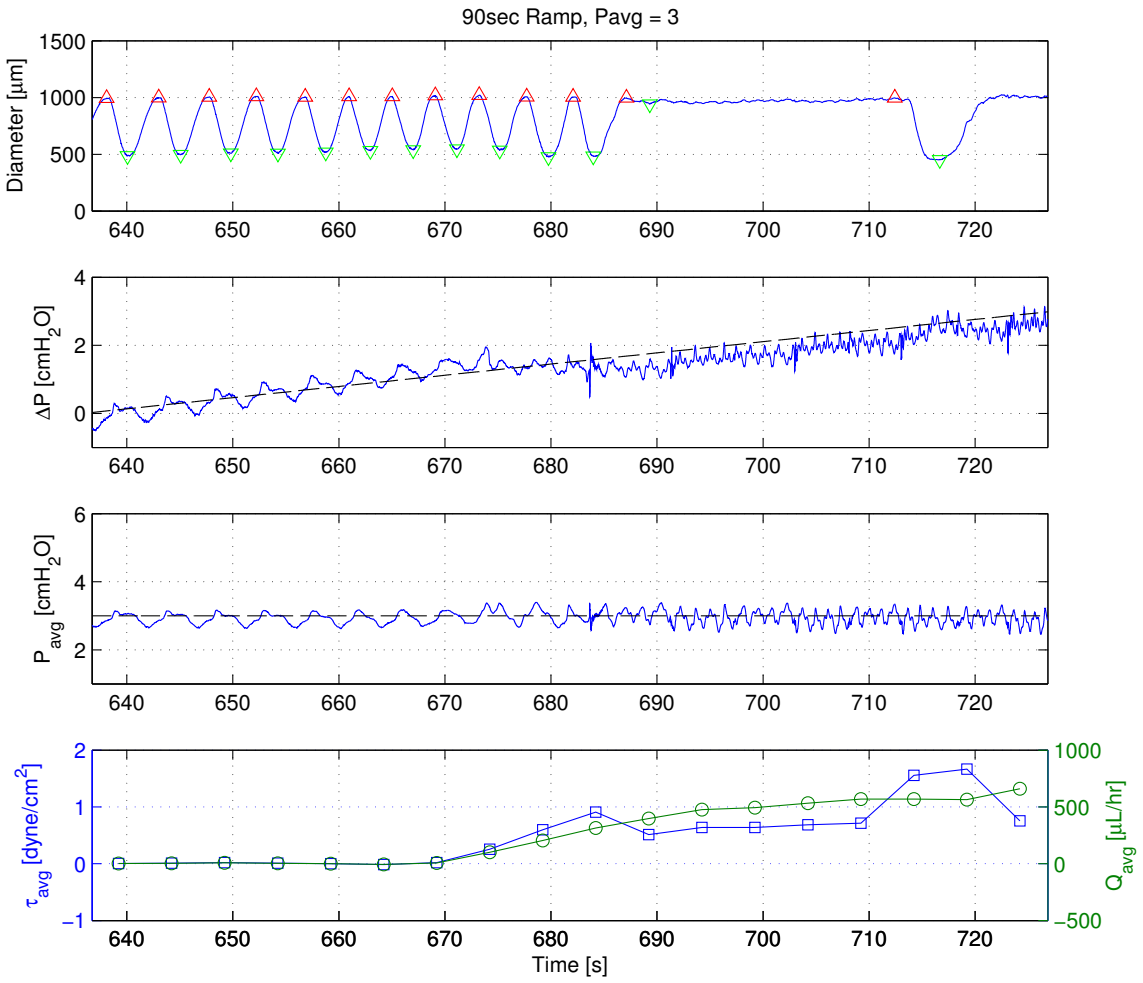


Figure C.12: 2013-08-28: 90-sec Ramp, $P_{\text{avg}} = 3$ cmH_2O.

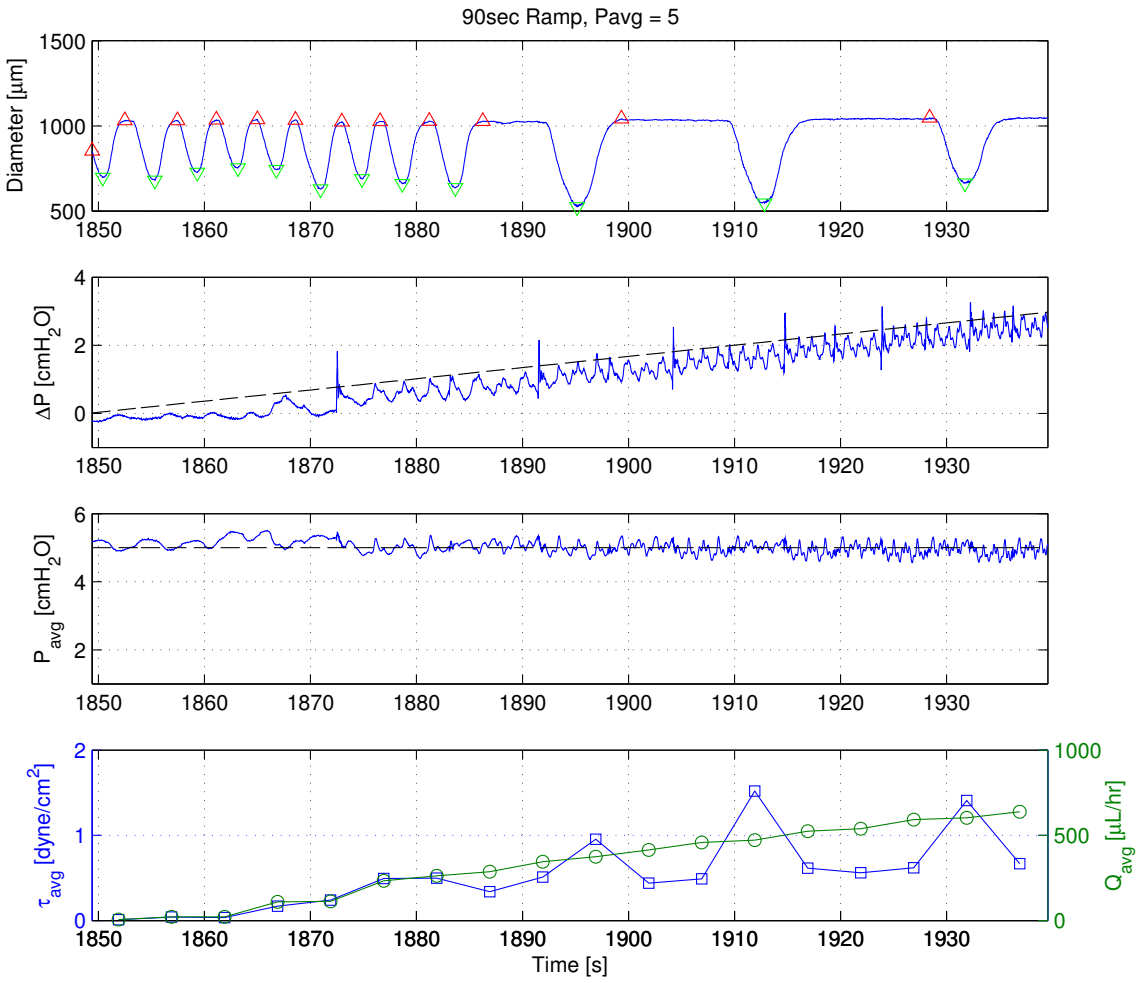


Figure C.13: 2013-08-28: 90-sec Ramp, $P_{\text{avg}} = 5$ cmH_2O.

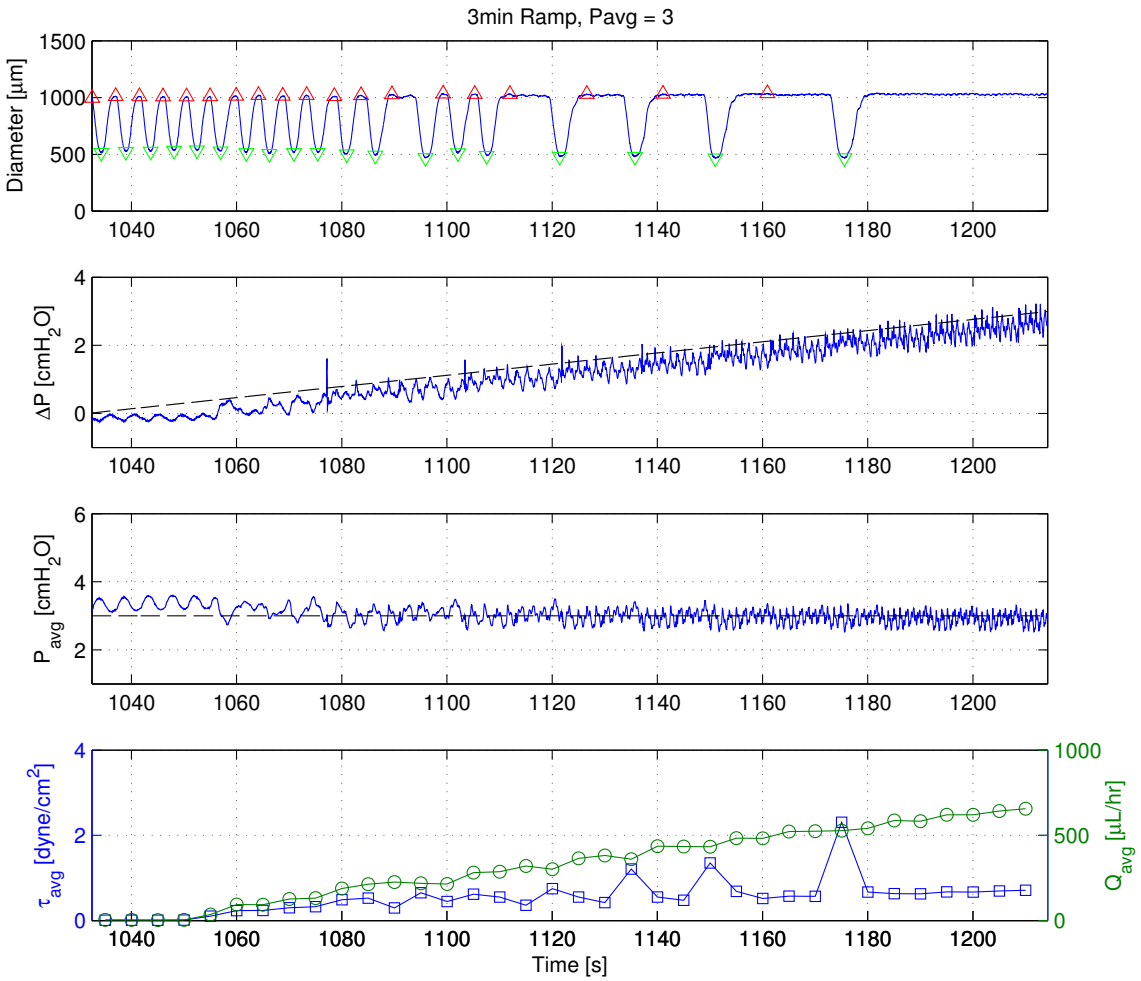


Figure C.14: 2013-08-28: 3-min Ramp, $P_{\text{avg}} = 3$ cmH_2O.

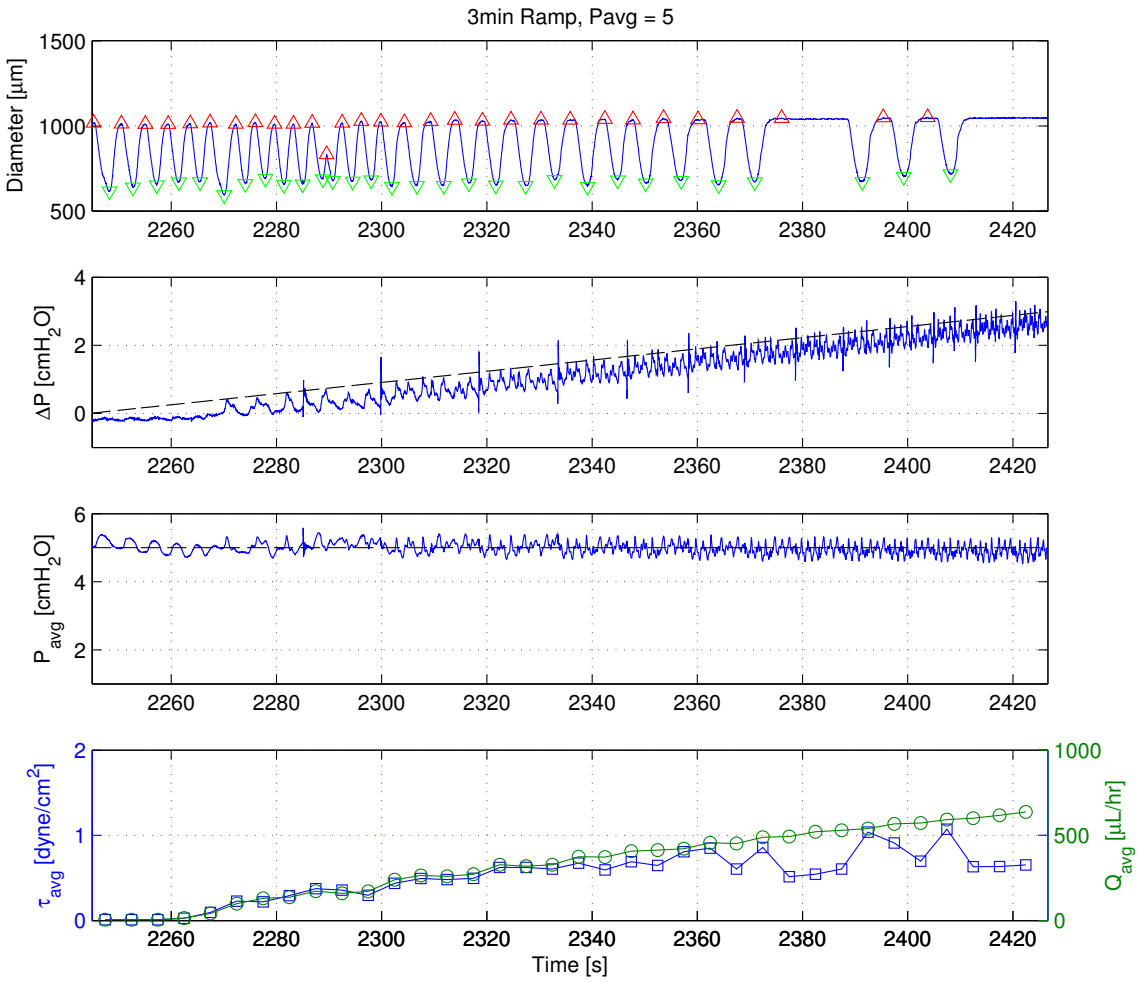


Figure C.15: 2013-08-28: 3-min Ramp, $P_{\text{avg}} = 5$ cmH_2O.

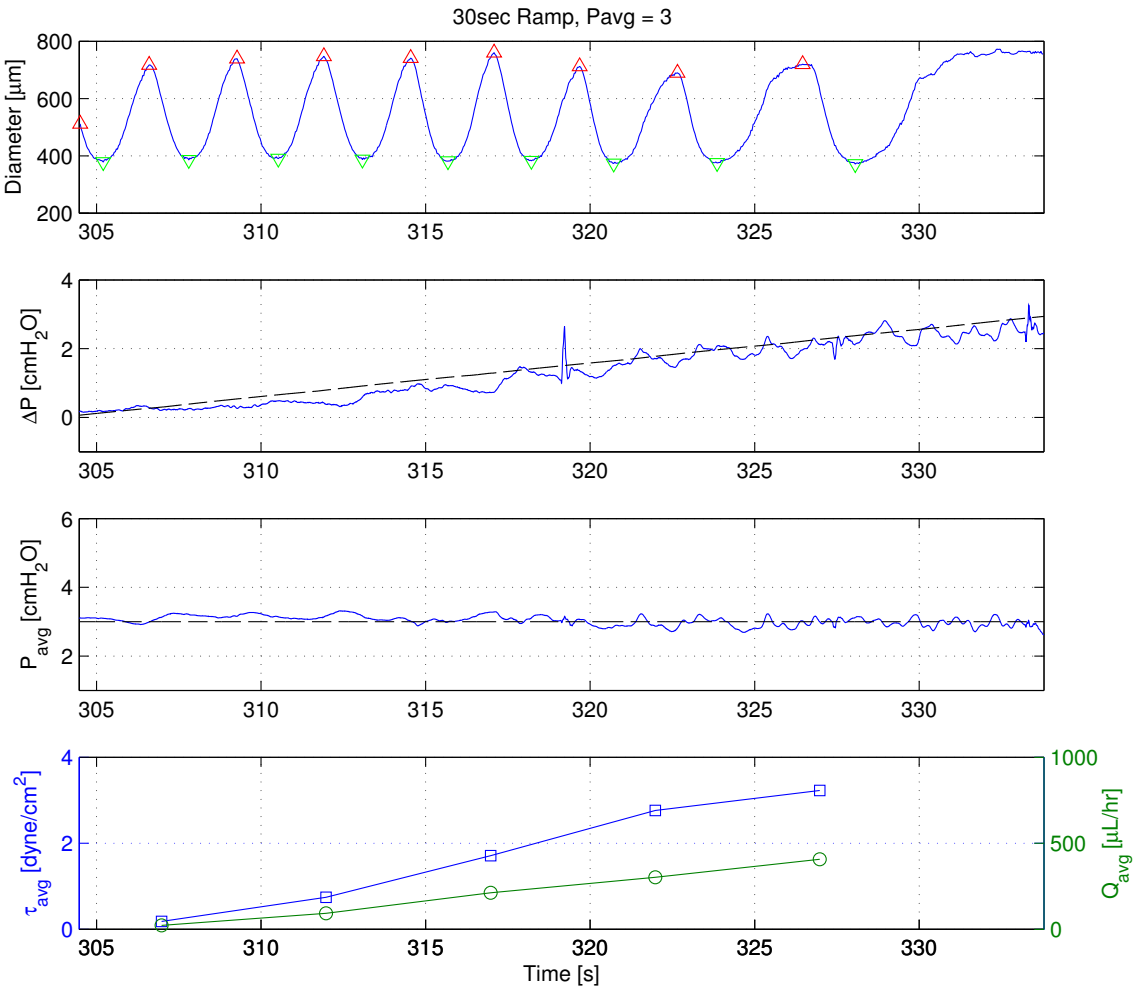


Figure C.16: 2013-08-29: 30-sec Ramp, $P_{\text{avg}} = 3$ cmH_2O.

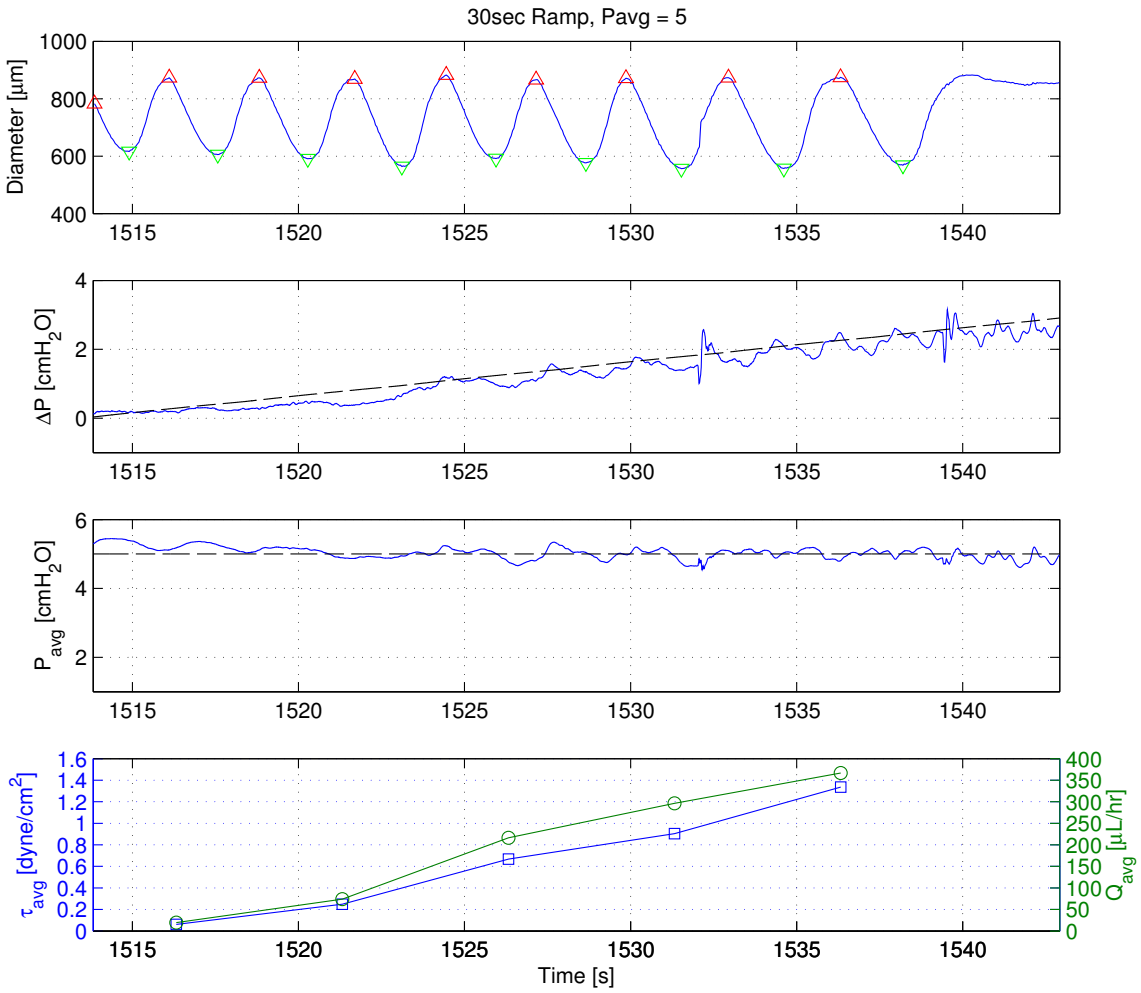


Figure C.17: 2013-08-29: 30-sec Ramp, $P_{\text{avg}} = 5$ cmH_2O.

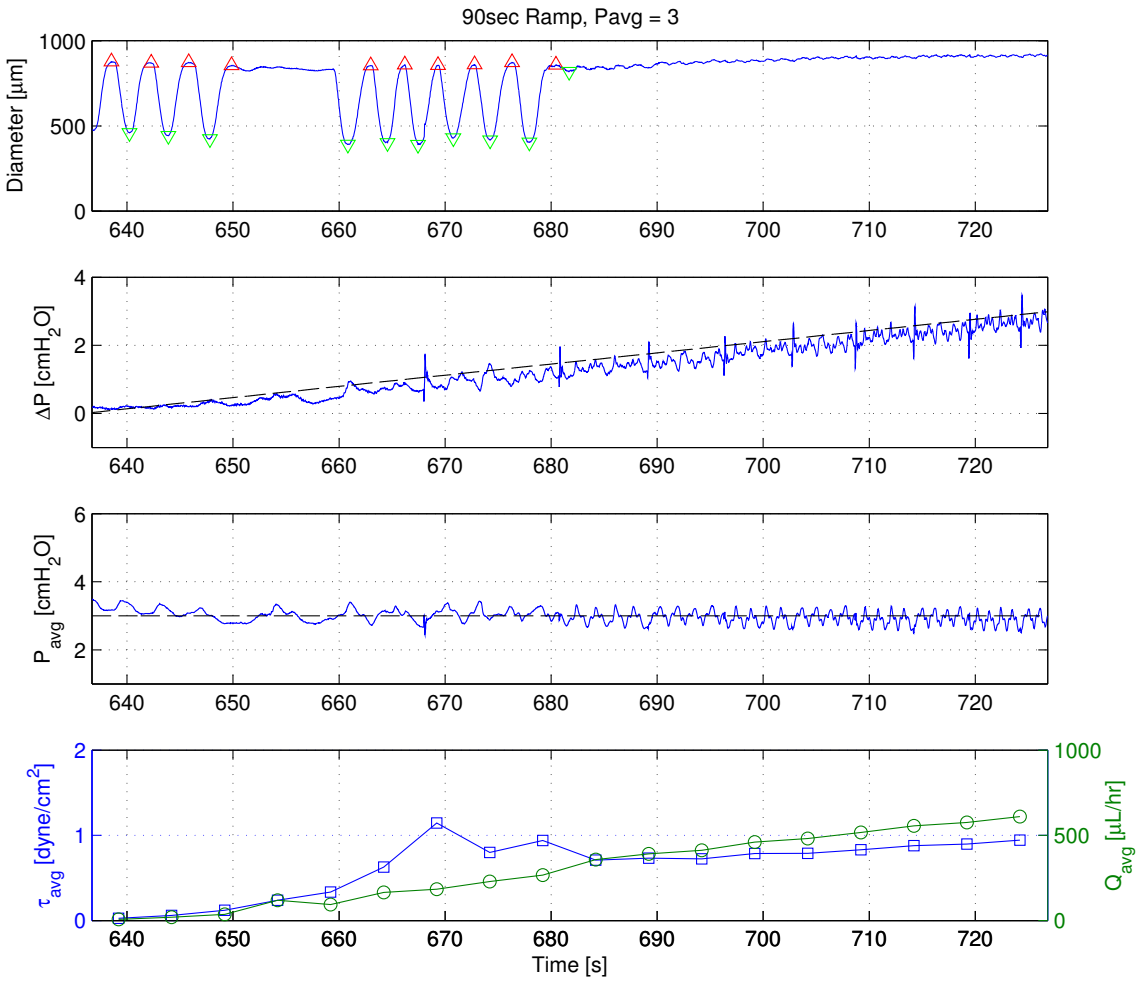


Figure C.18: 2013-08-29: 90-sec Ramp, $P_{\text{avg}} = 3$ cmH_2O.

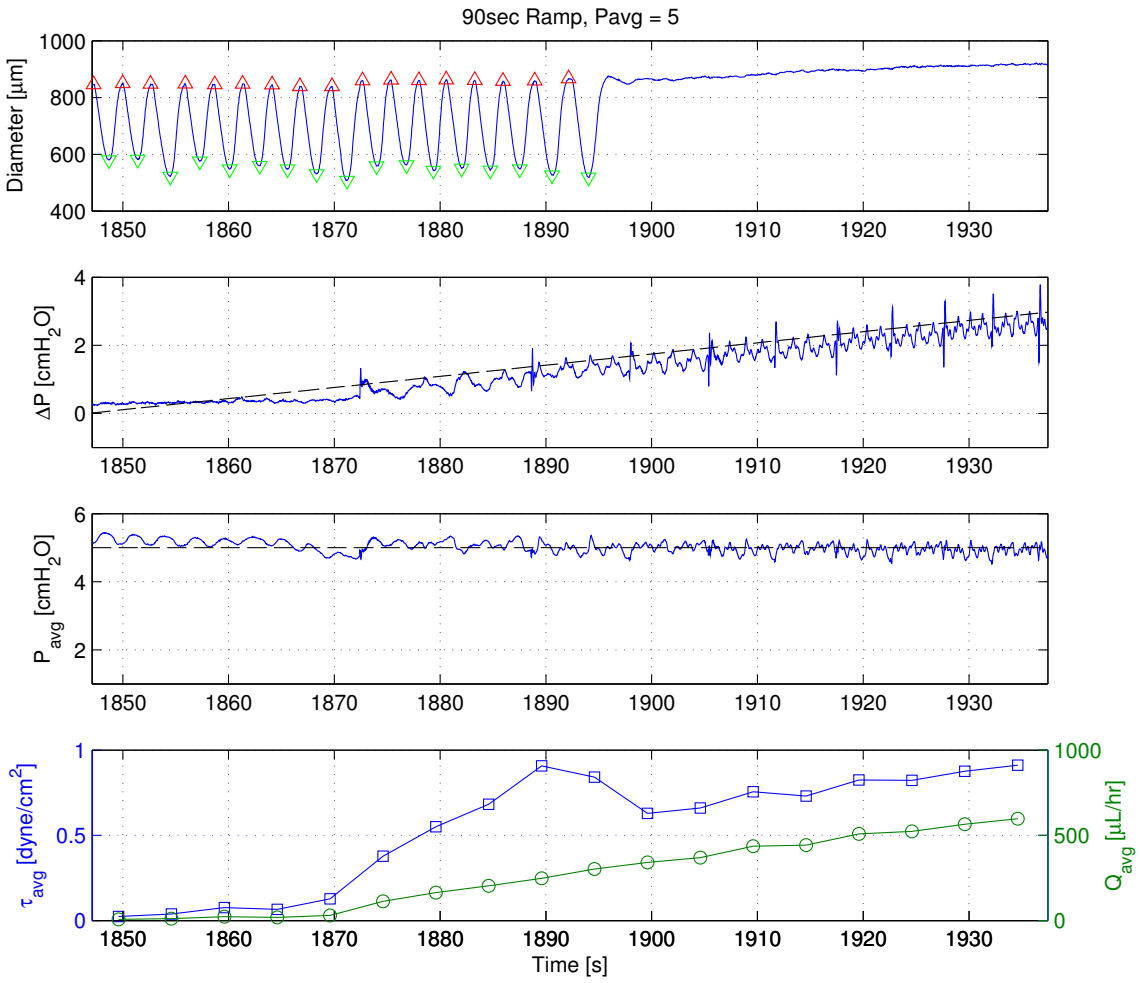


Figure C.19: 2013-08-29: 90-sec Ramp, $P_{\text{avg}} = 5$ cmH_2O.

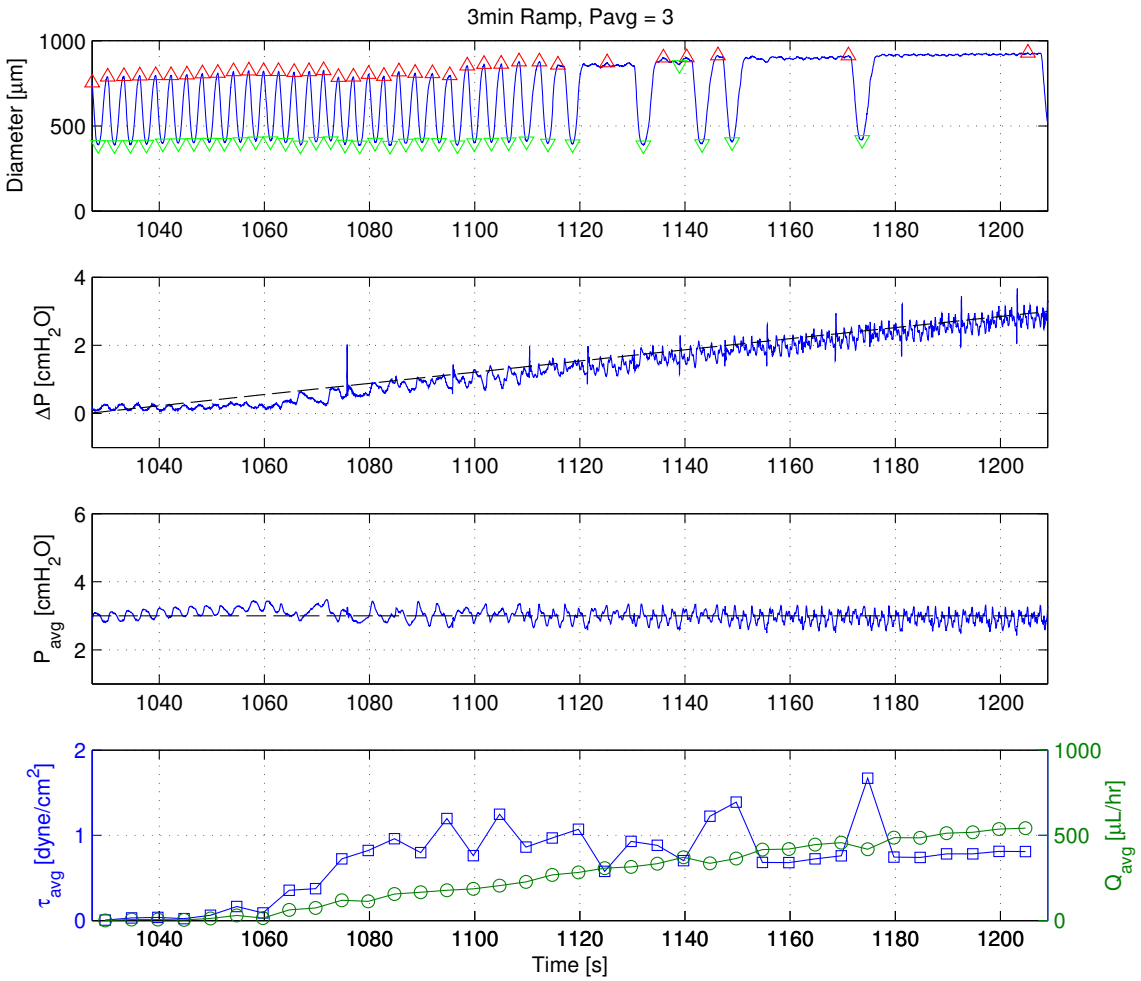


Figure C.20: 2013-08-29: 3-min Ramp, $P_{\text{avg}} = 3$ cmH_2O.

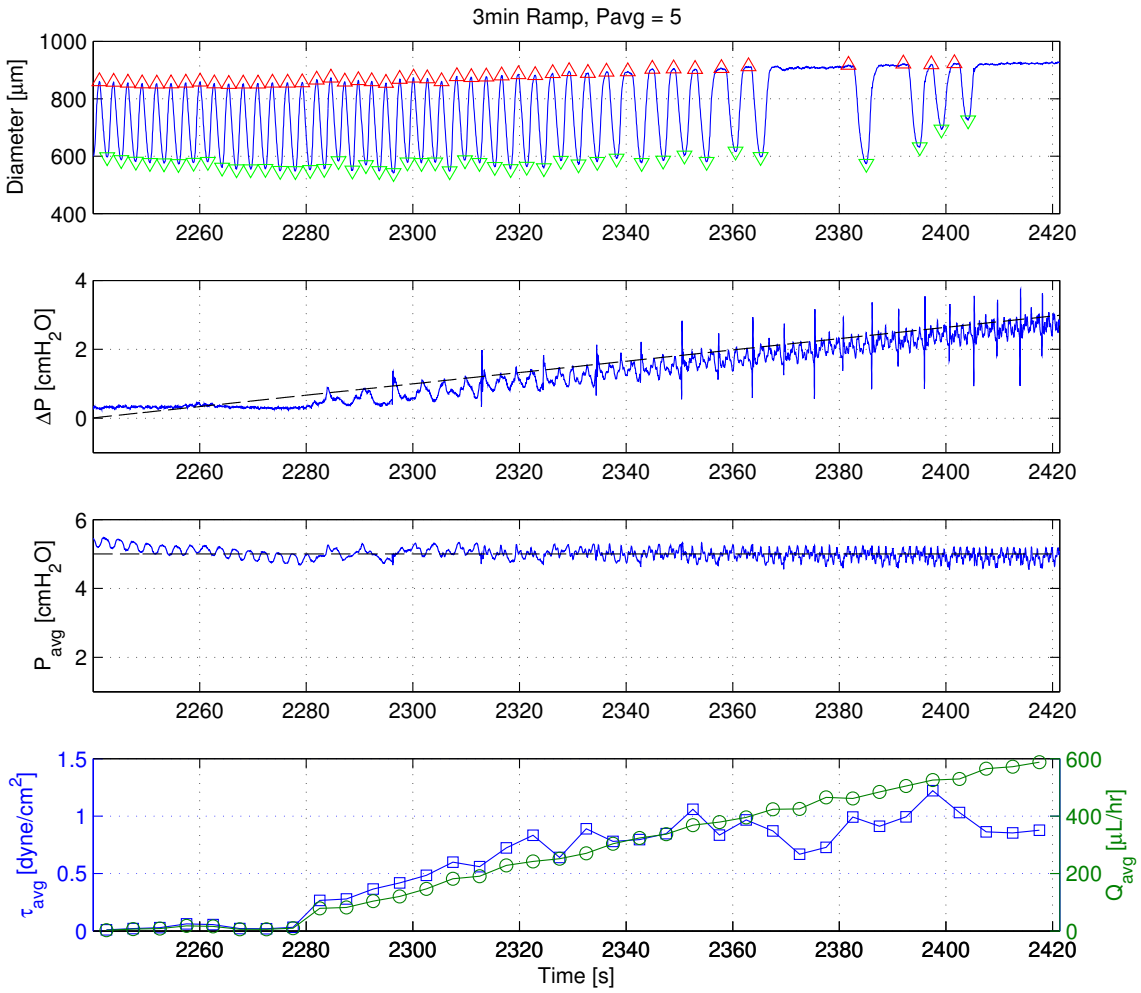


Figure C.21: 2013-08-29: 3-min Ramp, $P_{\text{avg}} = 5$ cmH_2O.

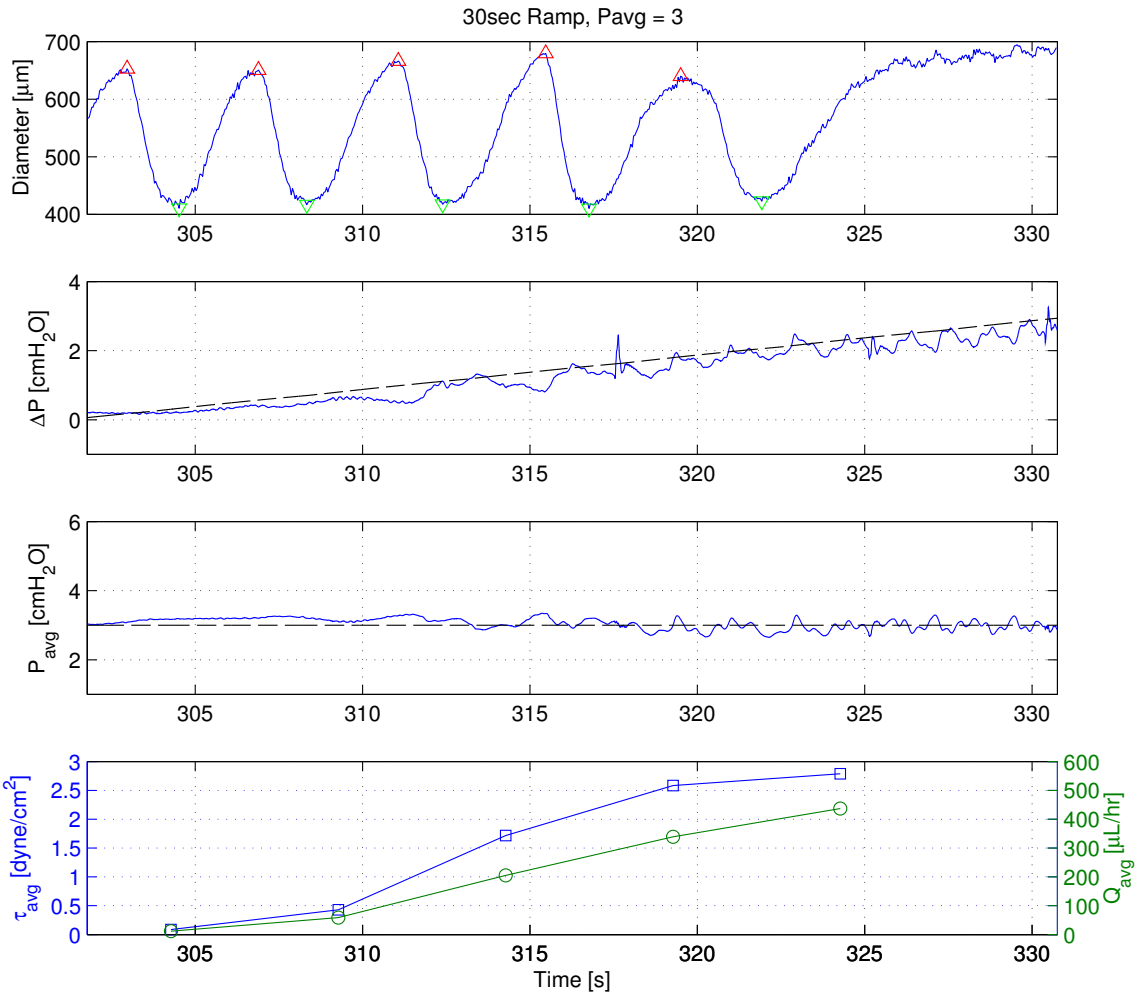


Figure C.22: 2013-09-03: 30-sec Ramp, $P_{\text{avg}} = 3$ cmH_2O.

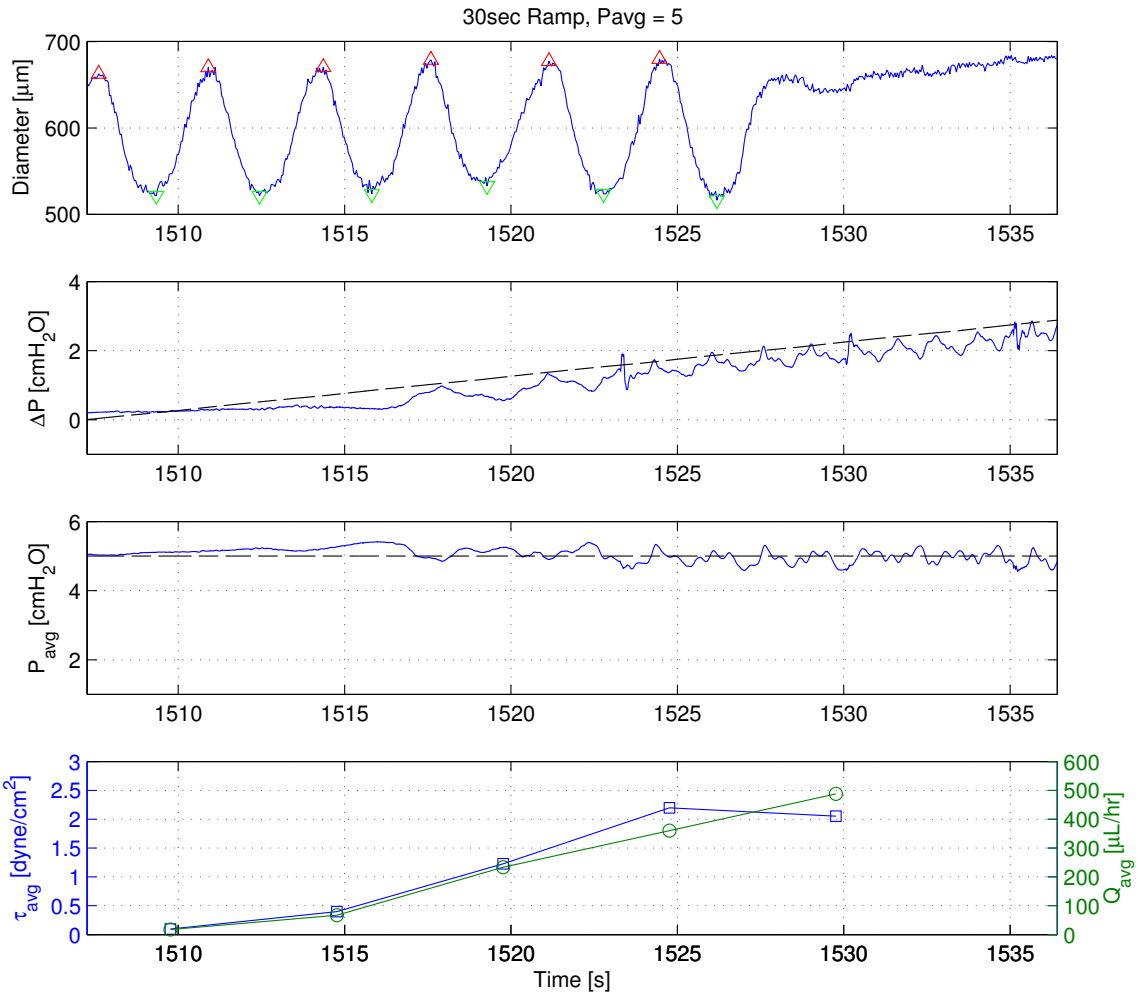


Figure C.23: 2013-09-03: 30-sec Ramp, $P_{\text{avg}} = 5$ cmH_2O.

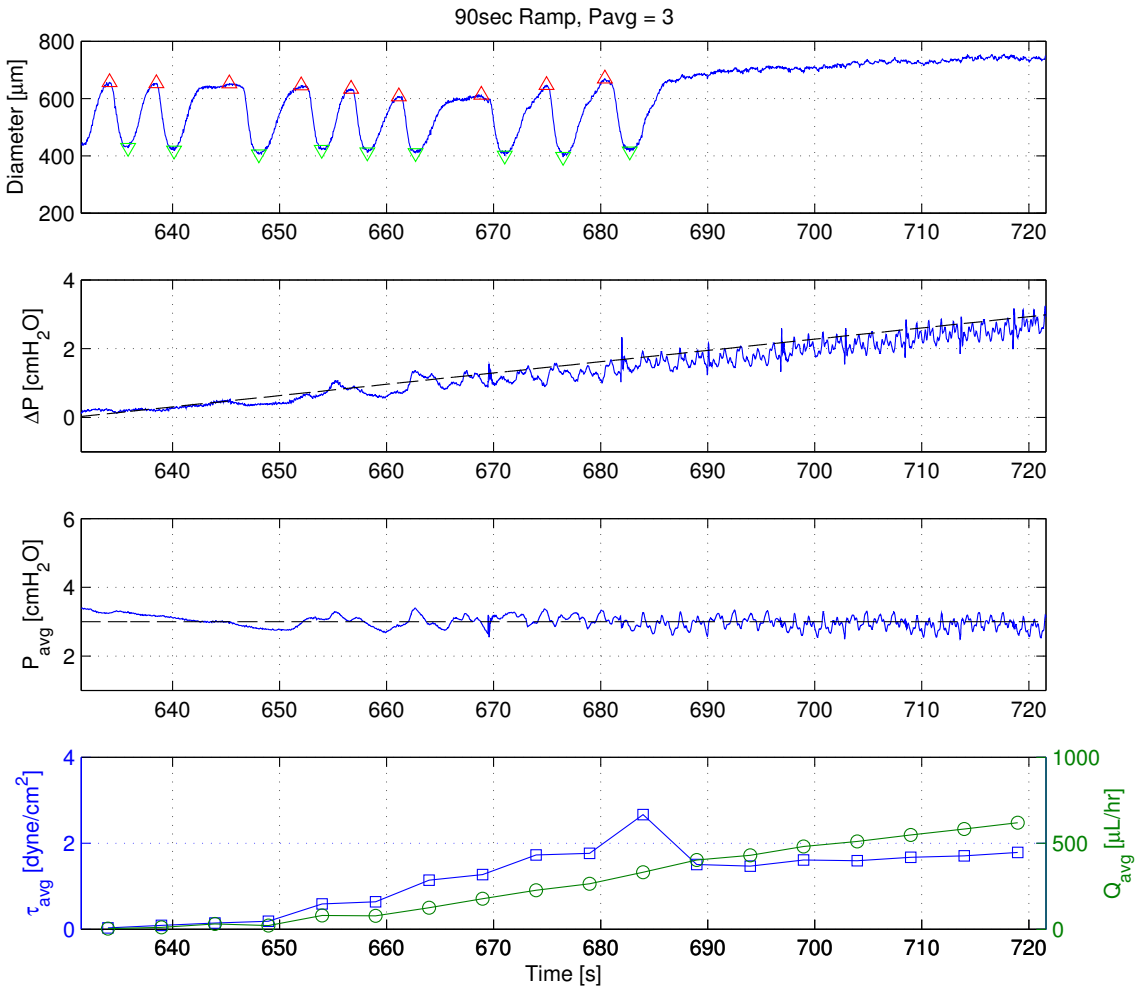


Figure C.24: 2013-09-03: 90-sec Ramp, $P_{\text{avg}} = 3$ cmH_2O.

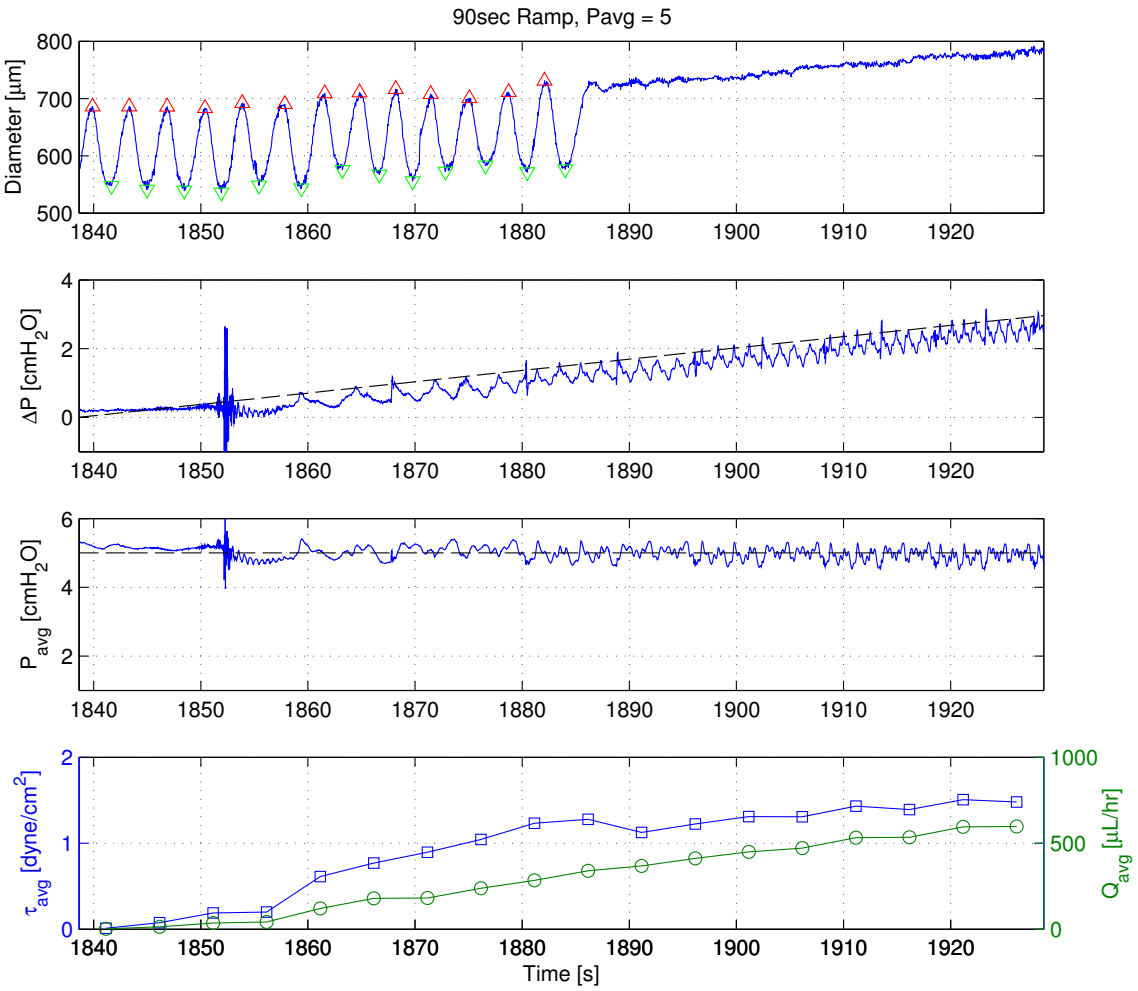


Figure C.25: 2013-09-03: 90-sec Ramp, $P_{\text{avg}} = 5$ cmH_2O.

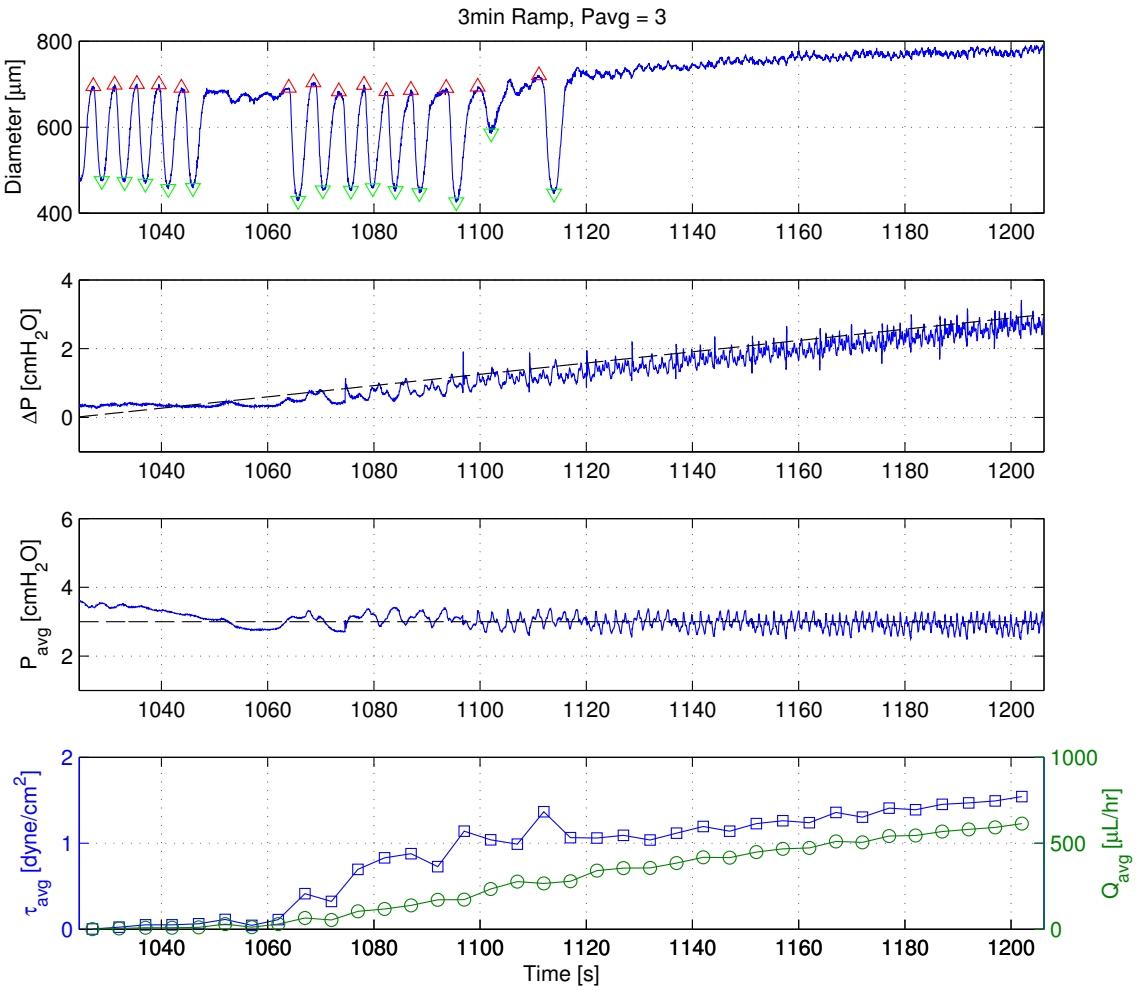


Figure C.26: 2013-09-03: 3-min Ramp, $P_{\text{avg}} = 3$ cmH_2O.

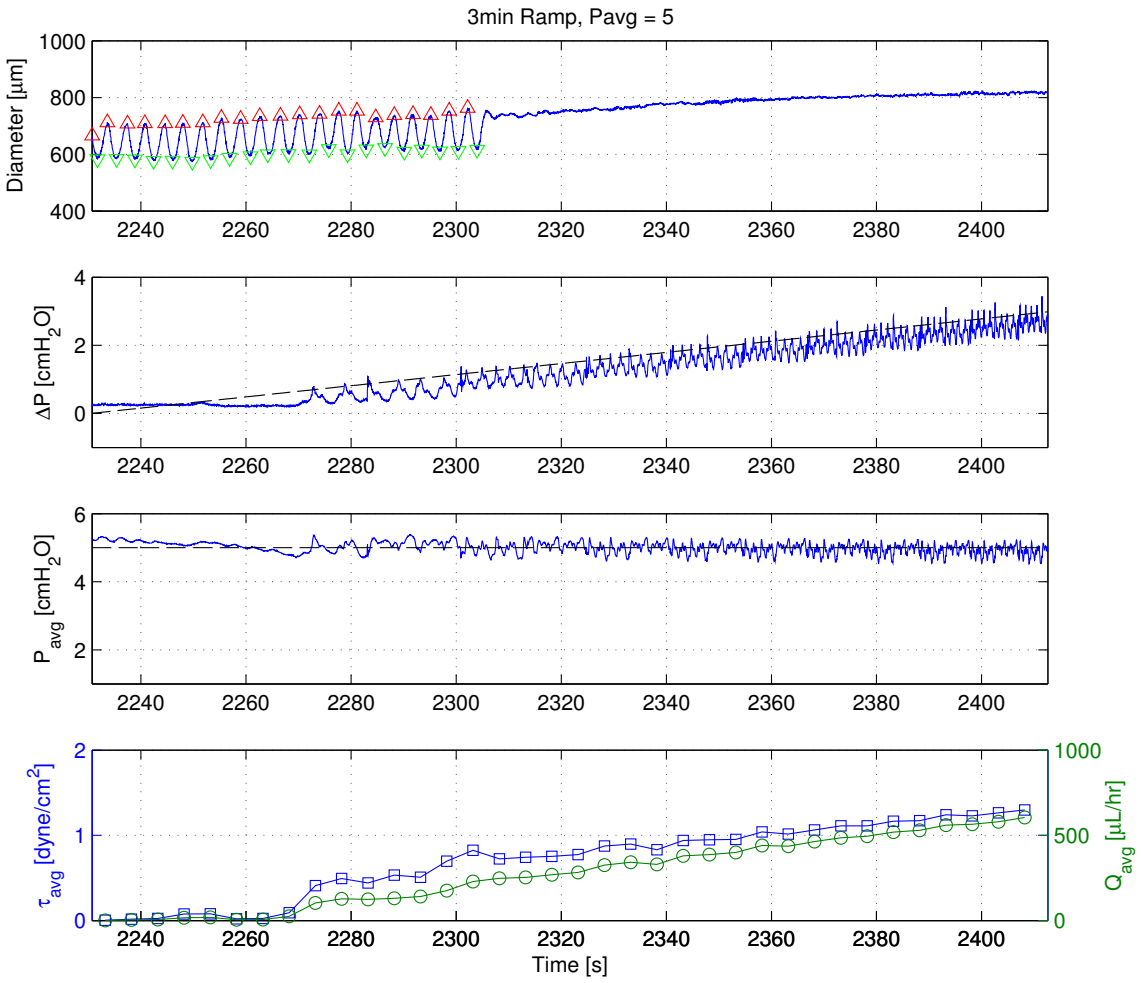


Figure C.27: 2013-09-03: 3-min Ramp, $P_{\text{avg}} = 5$ cmH_2O.

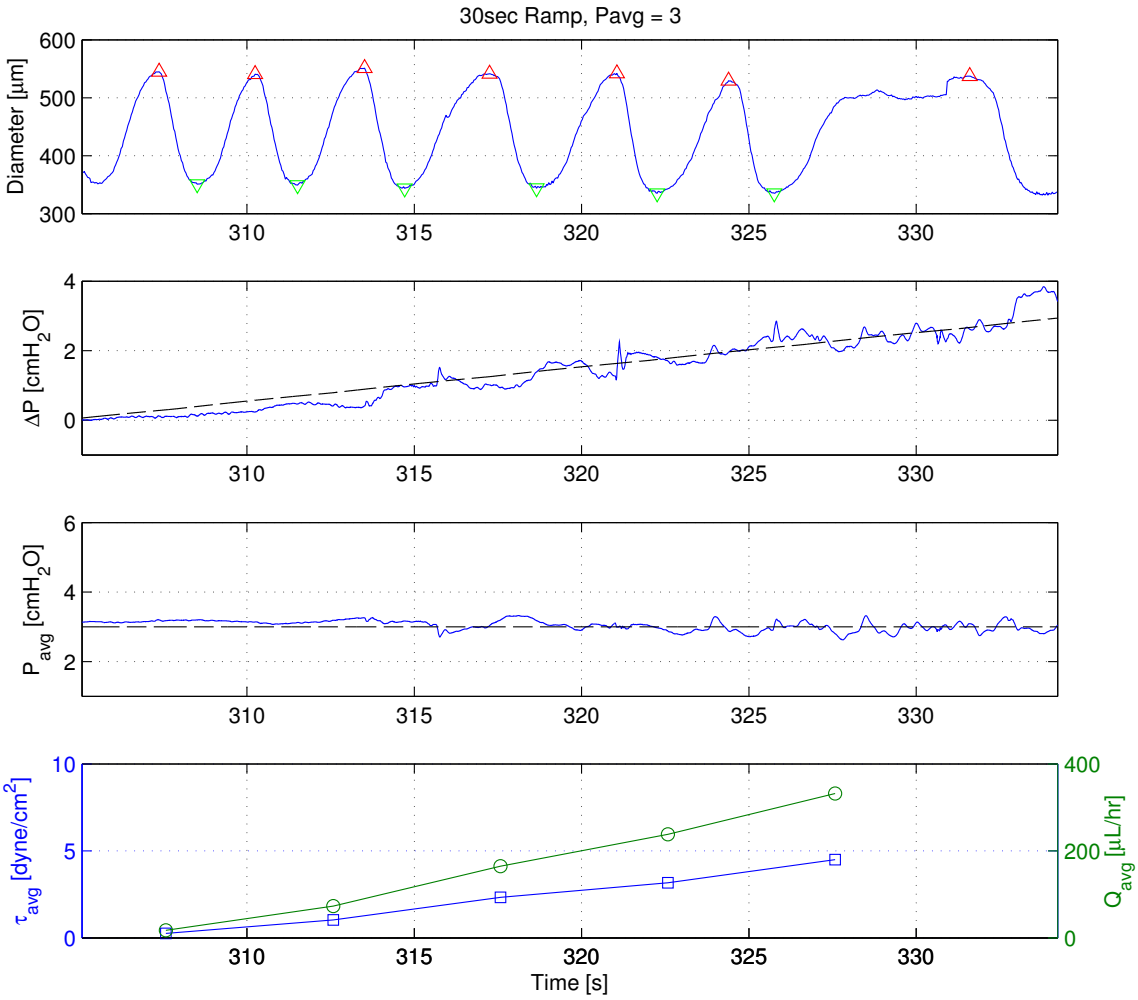


Figure C.28: 2013-09-04: 30-sec Ramp, $P_{\text{avg}} = 3$ cmH_2O.

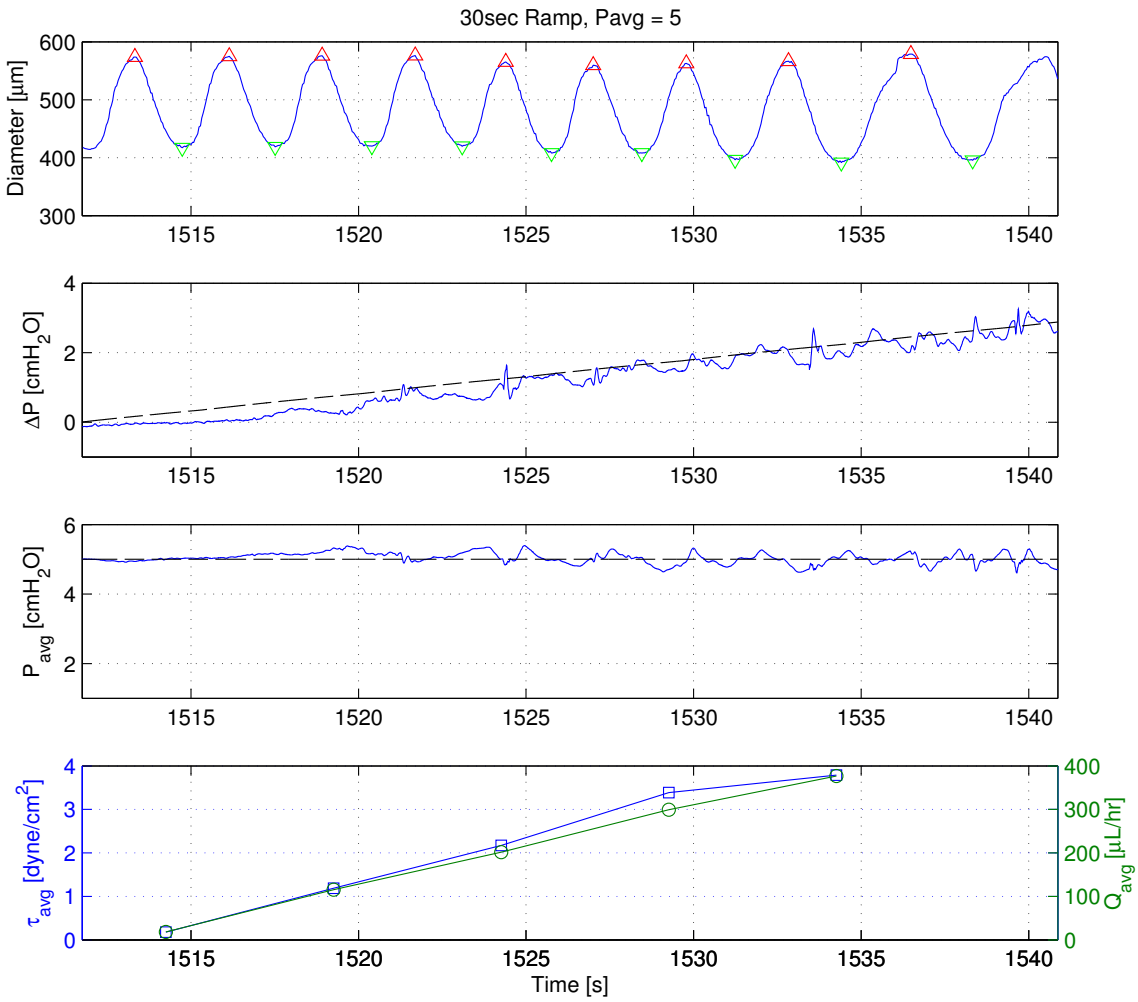


Figure C.29: 2013-09-04: 30-sec Ramp, $P_{\text{avg}} = 5$ cmH_2O.

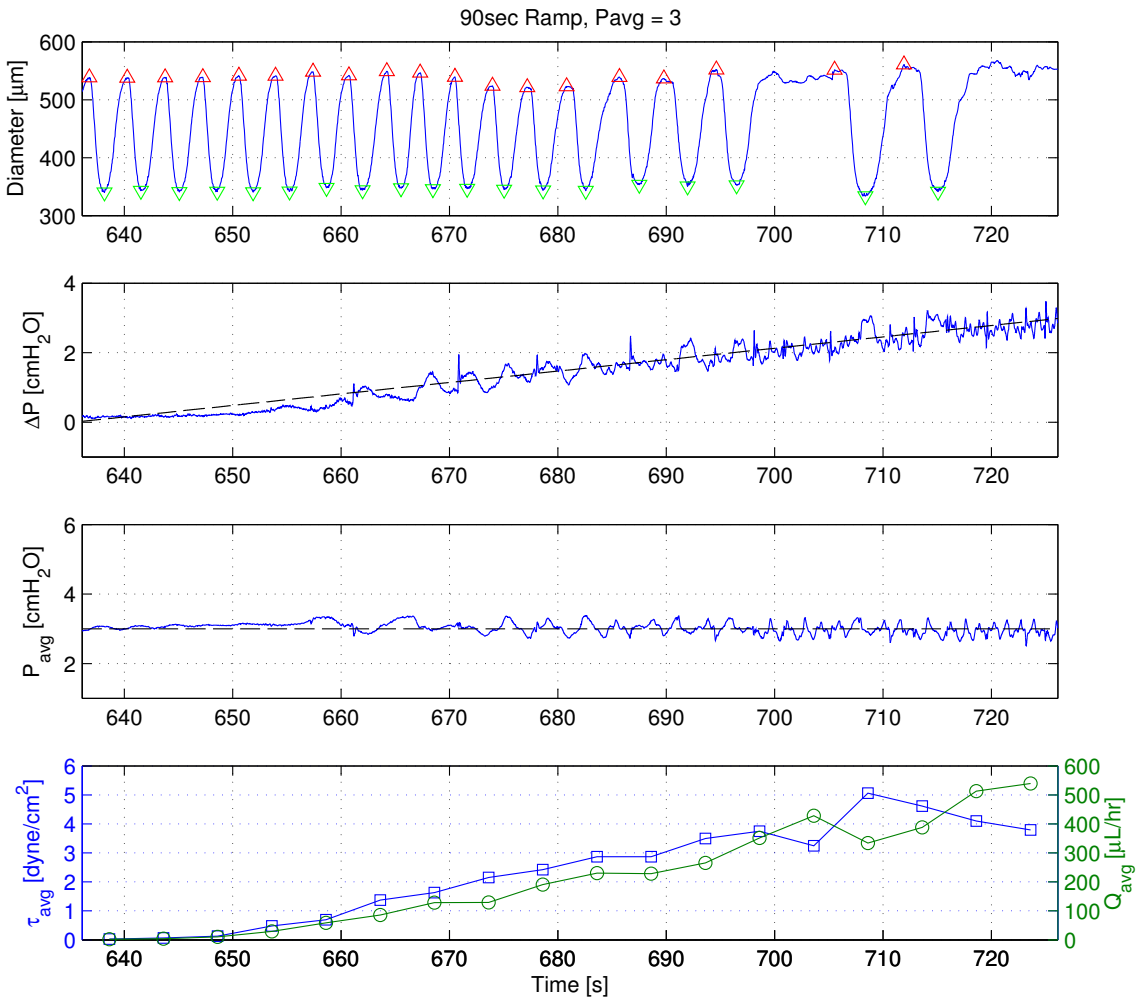


Figure C.30: 2013-09-04: 90-sec Ramp, $P_{\text{avg}} = 3$ cmH_2O.

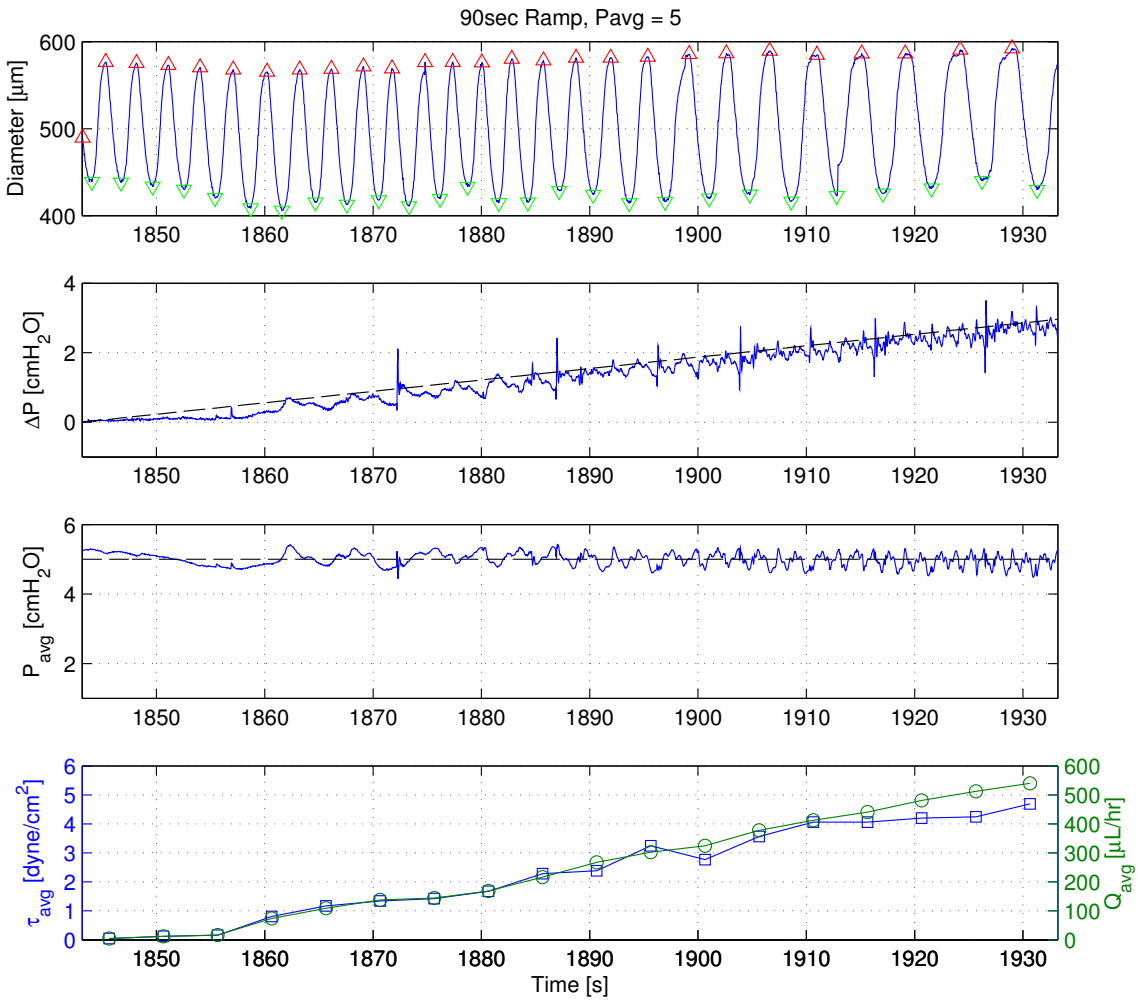


Figure C.31: 2013-09-04: 90-sec Ramp, $P_{\text{avg}} = 5$ cmH_2O.

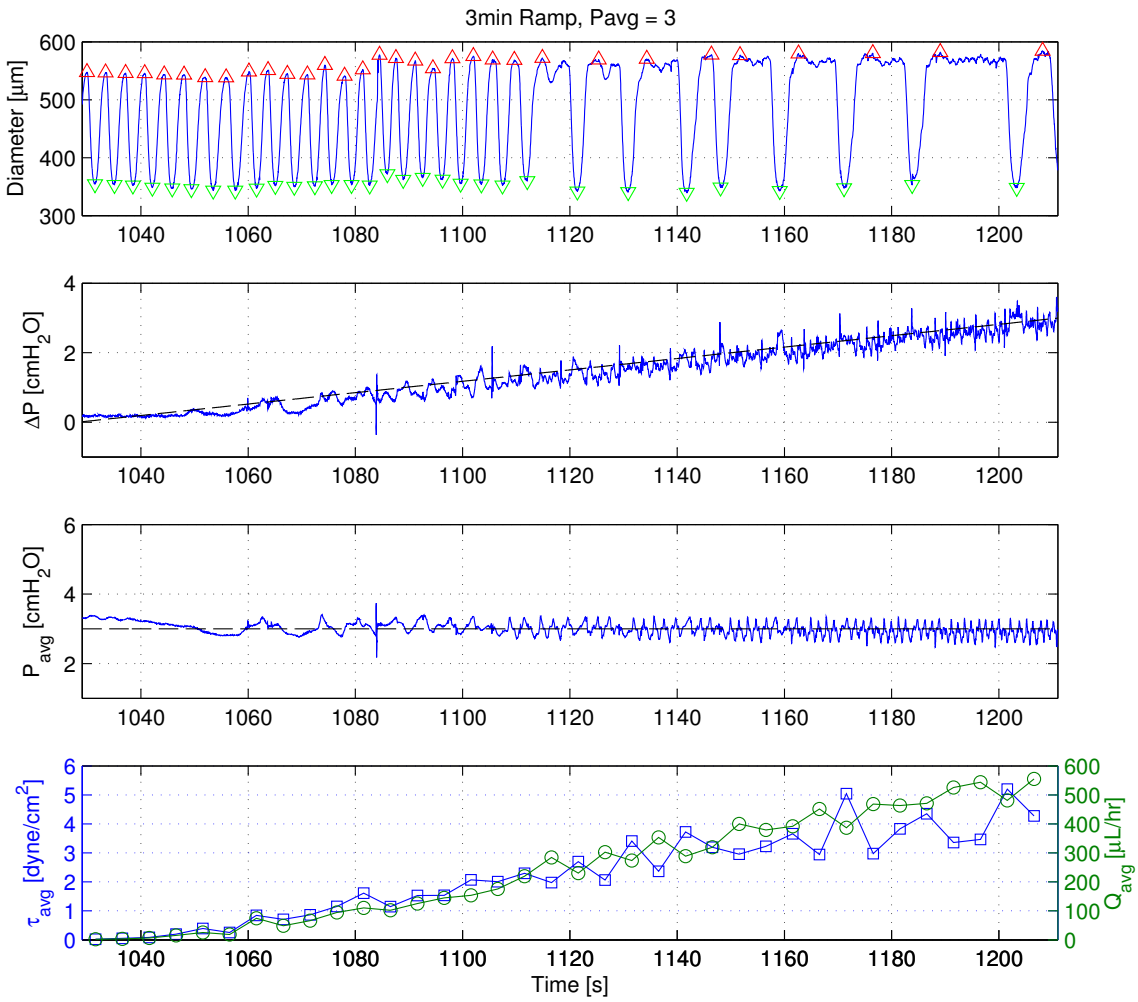


Figure C.32: 2013-09-04: 3-min Ramp, $P_{\text{avg}} = 3$ cmH_2O.

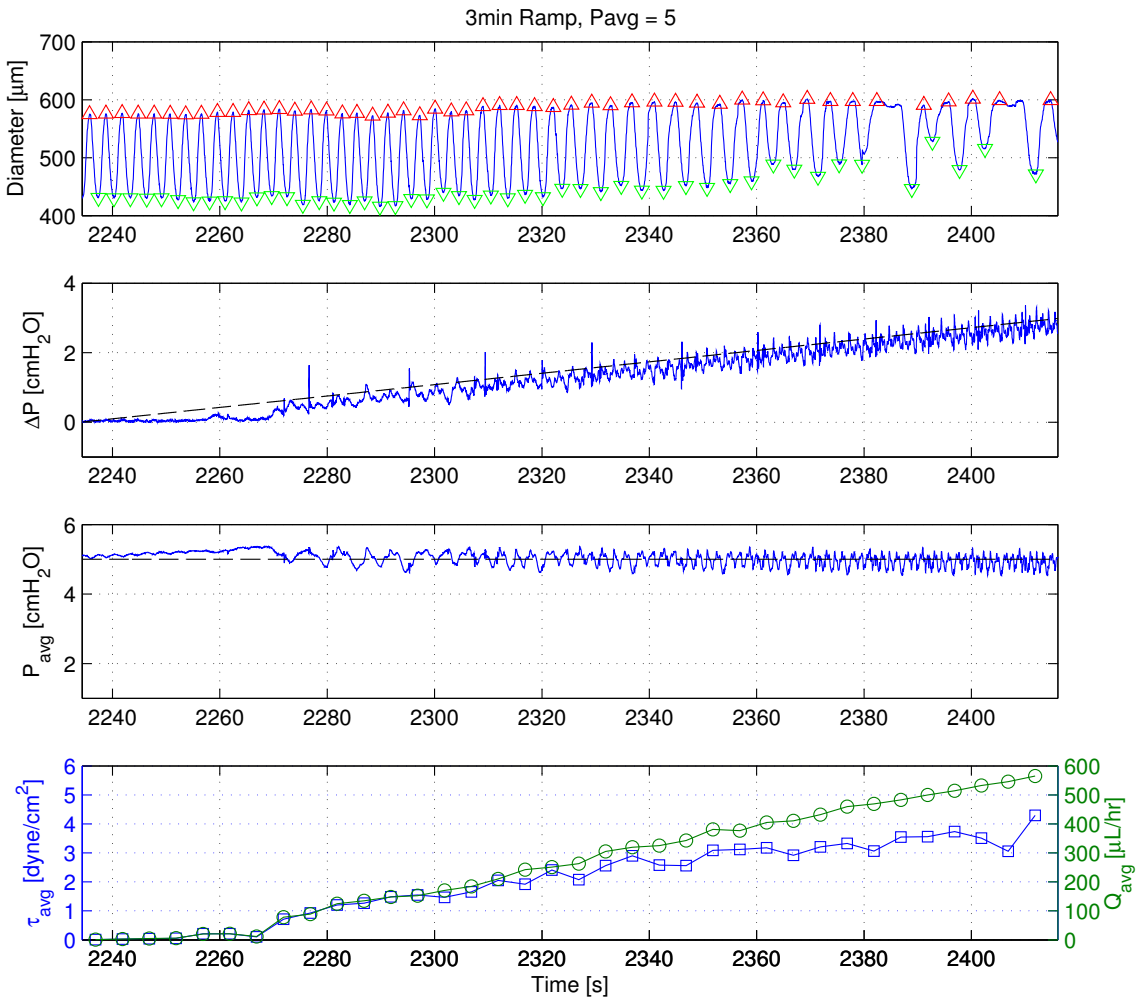


Figure C.33: 2013-09-04: 3-min Ramp, $P_{\text{avg}} = 5$ cmH_2O.

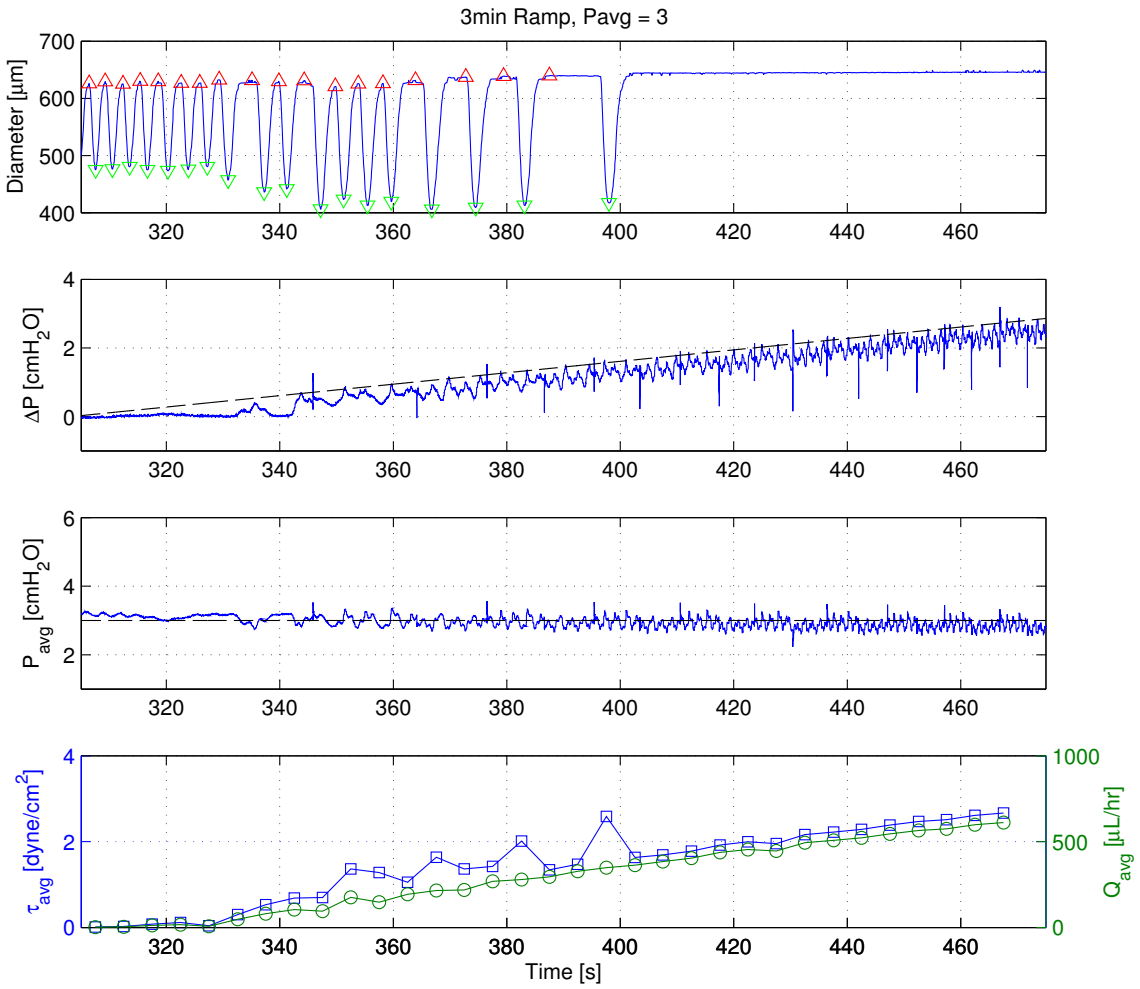


Figure C.34: 2014-03-12: 3-min Ramp, $P_{\text{avg}} = 3$ cmH_2O.

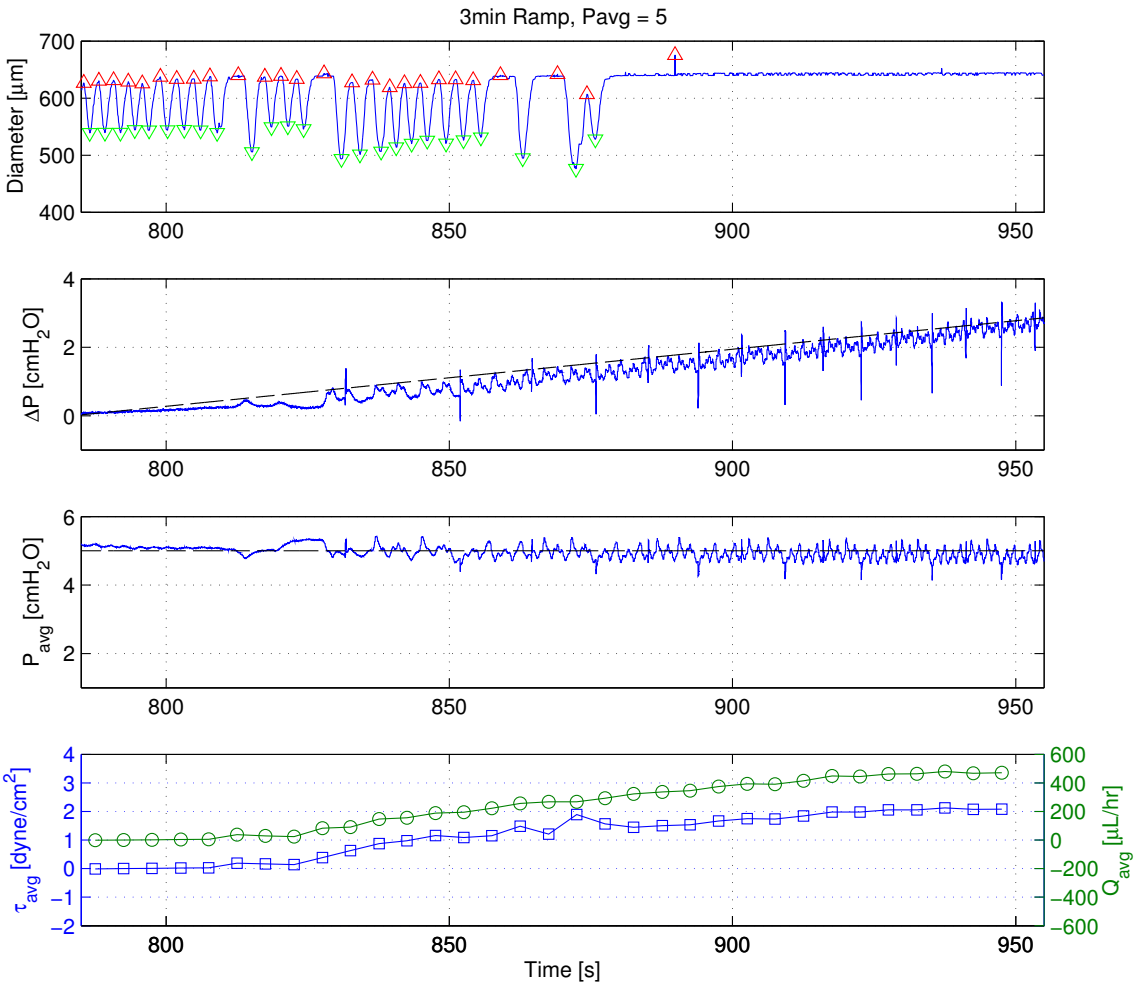


Figure C.35: 2014-03-12: 3-min Ramp, $P_{\text{avg}} = 5$ cmH_2O.

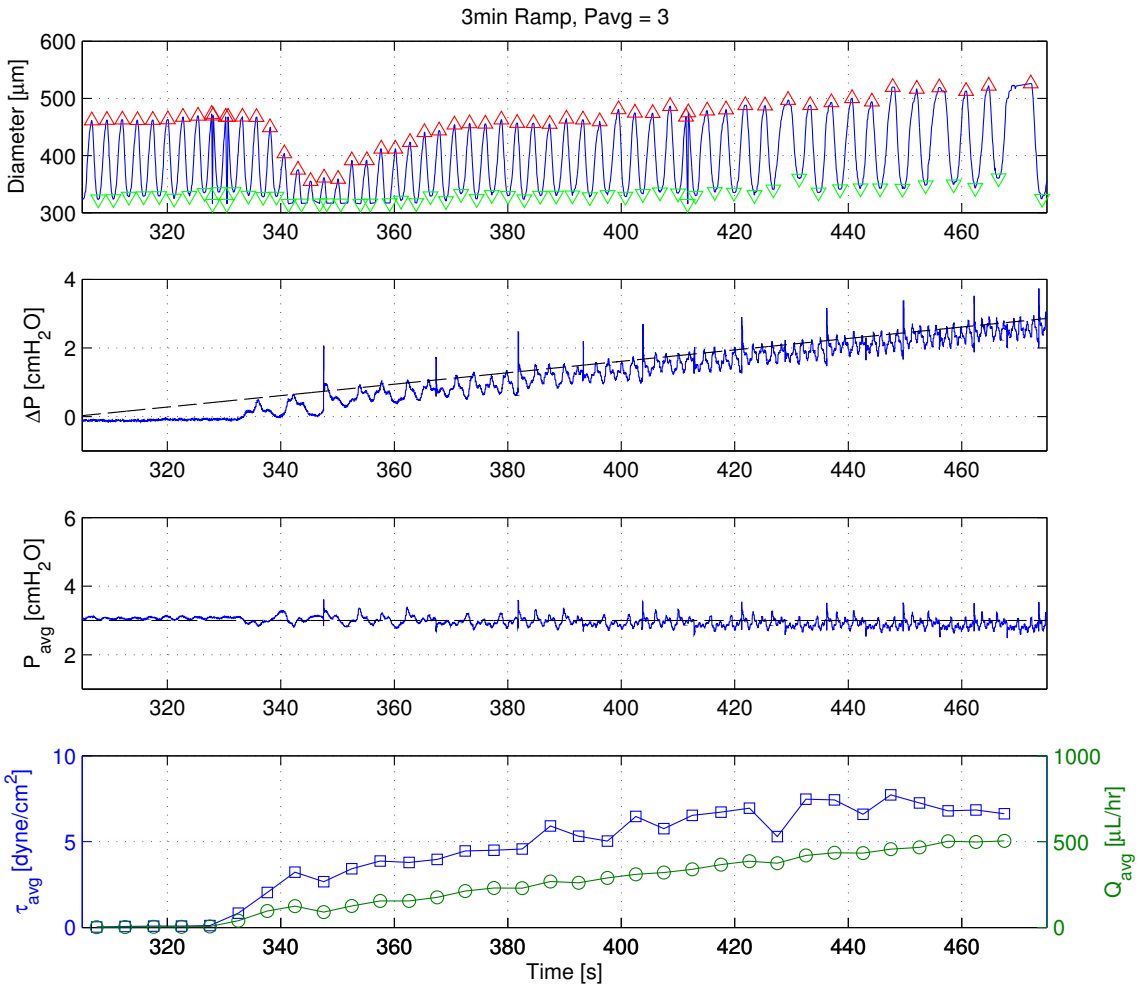


Figure C.36: 2014-03-18: 3-min Ramp, $P_{\text{avg}} = 3$ cmH_2O.

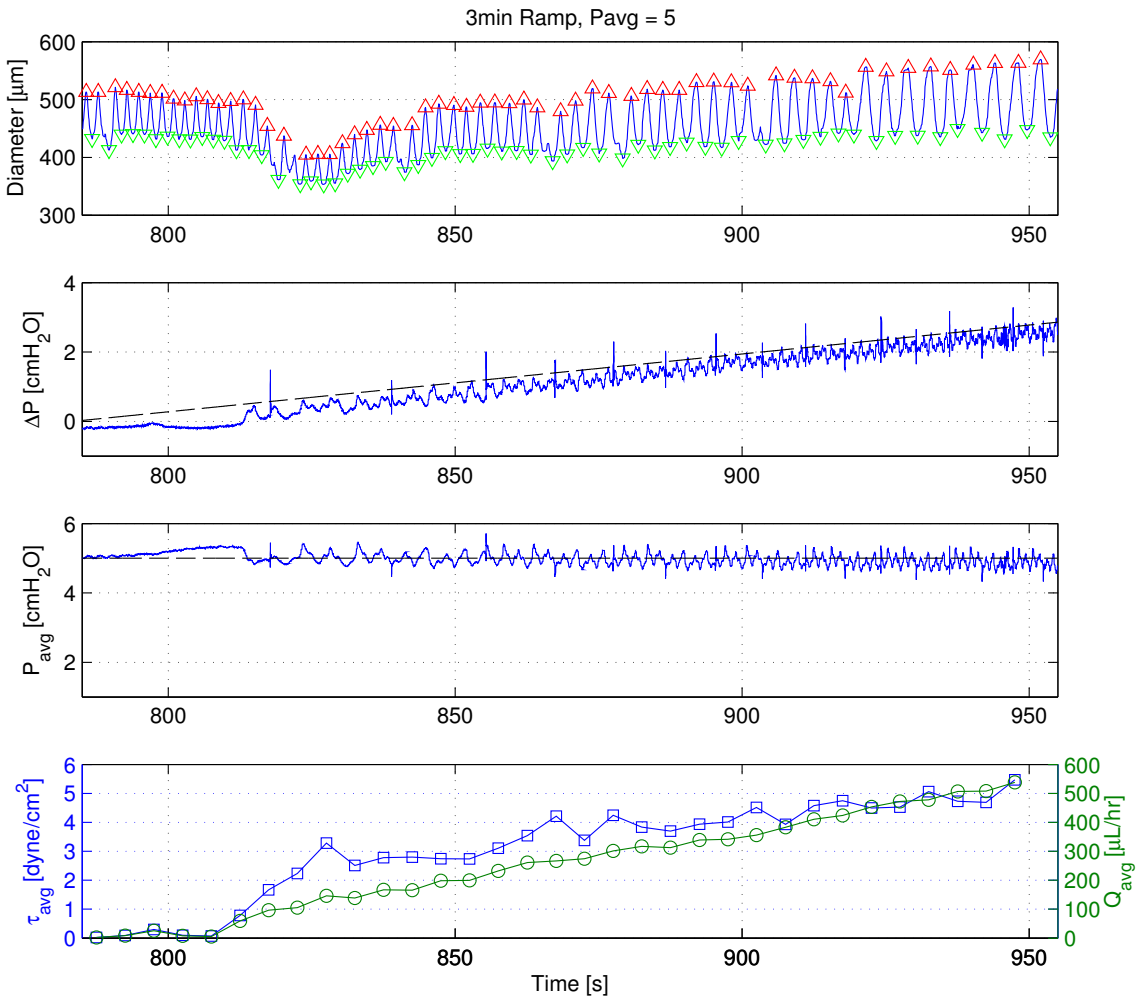


Figure C.37: 2014-03-18: 3-min Ramp, $P_{\text{avg}} = 5$ cmH_2O.

C.4 Sine Wave Data for Individual Vessels

Note: Sine conditions with a zero mean wall shear stress experienced an air bubble trapped in the ELPS tubing; thus, they should not be used in any subsequent analysis.

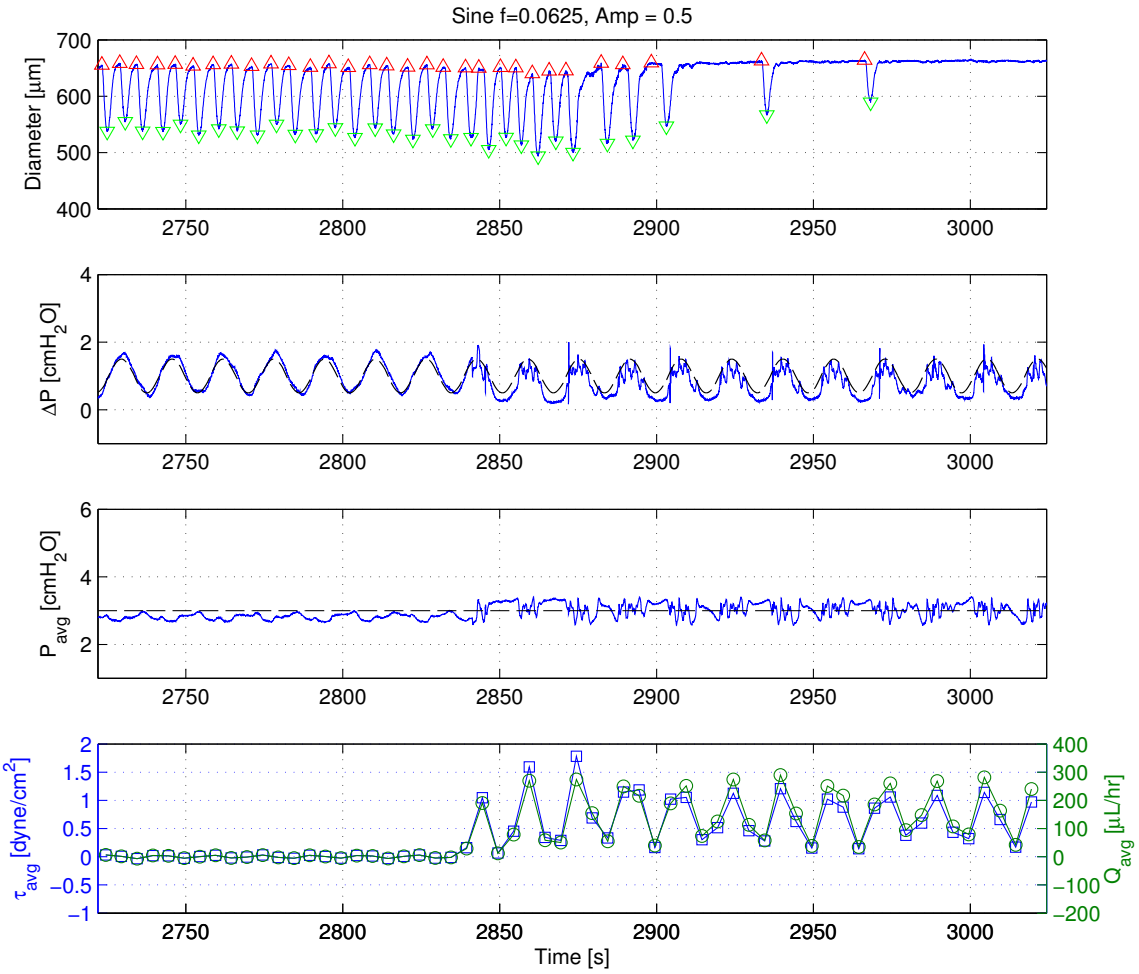


Figure C.38: 2013-08-27: 5-min Sine, $A = 0.5$ cmH_2O, $f = 0.0625$ Hz.

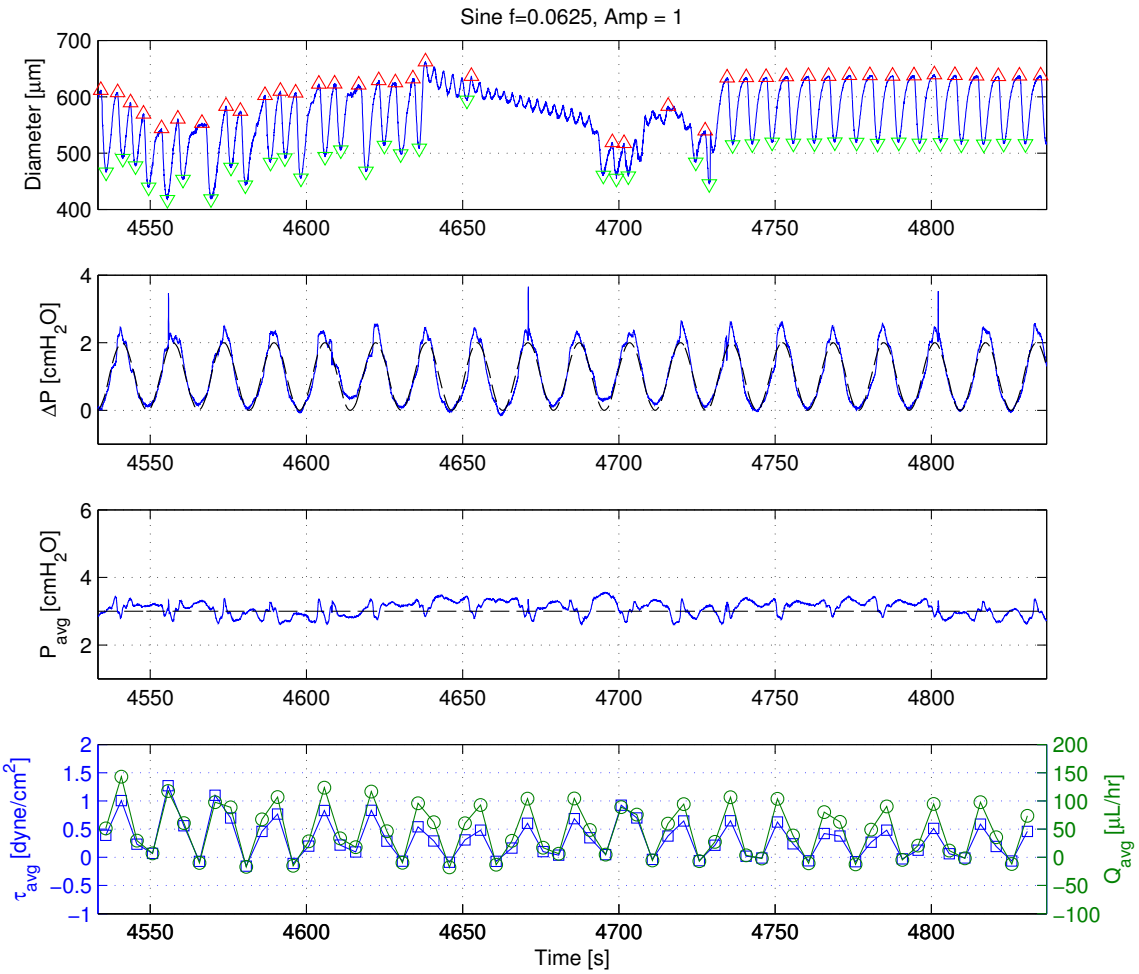


Figure C.39: 2013-08-27: 5-min Sine, $A = 1$ cmH$_2$O, $f = 0.0625$ Hz.

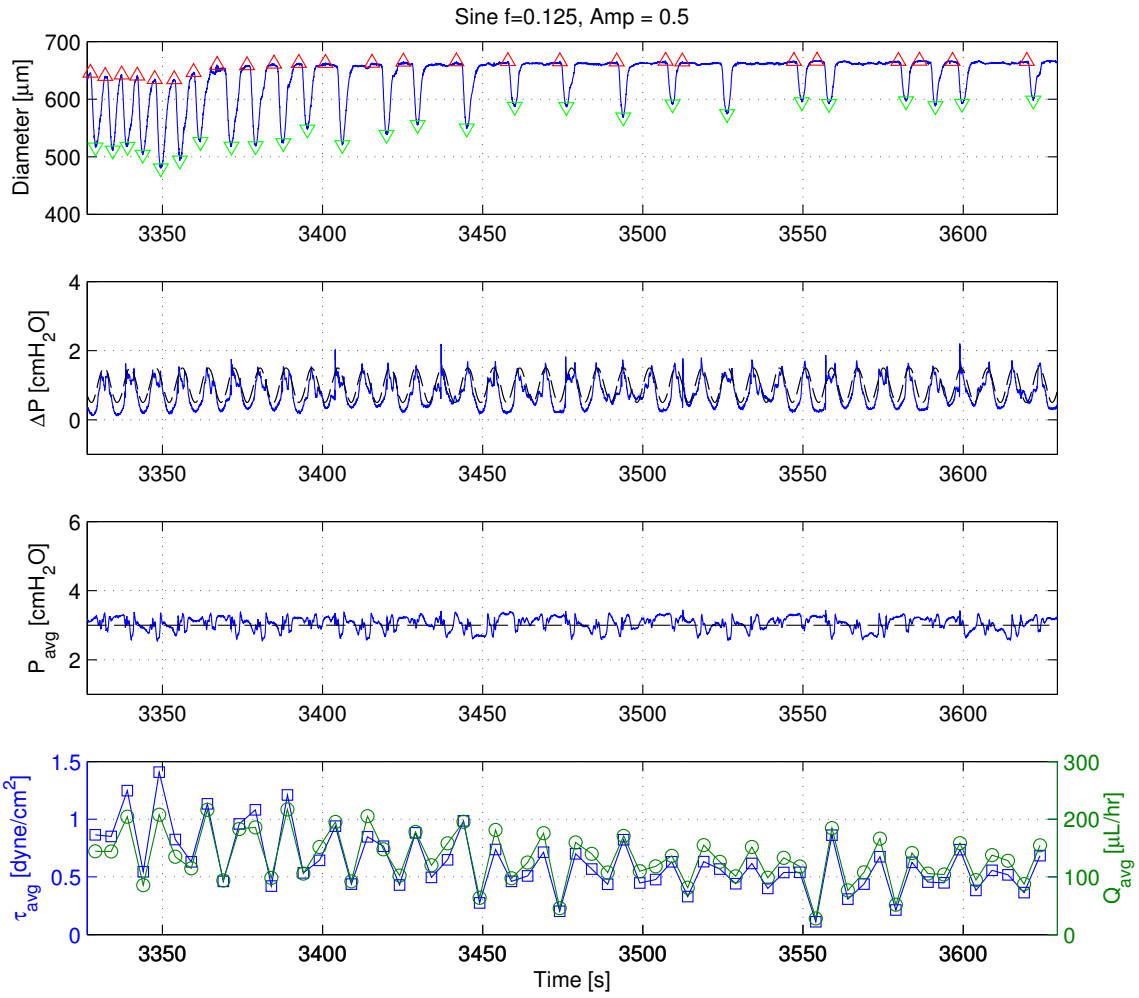


Figure C.40: 2013-08-27: 5-min Sine, $A = 0.5$ cmH$_2$O, $f = 0.125$ Hz.

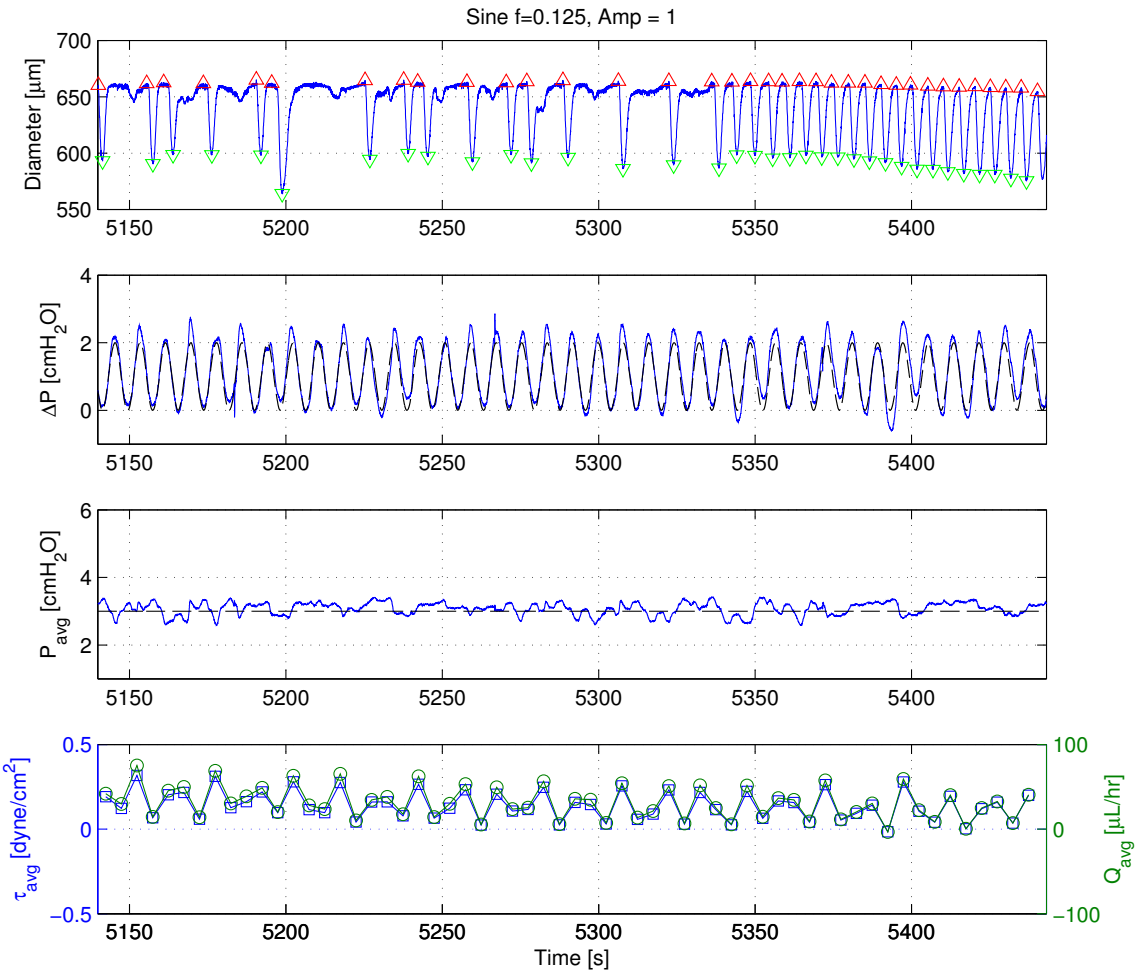


Figure C.41: 2013-08-27: 5-min Sine, $A = 1$ cmH$_2$O, $f = 0.125$ Hz.

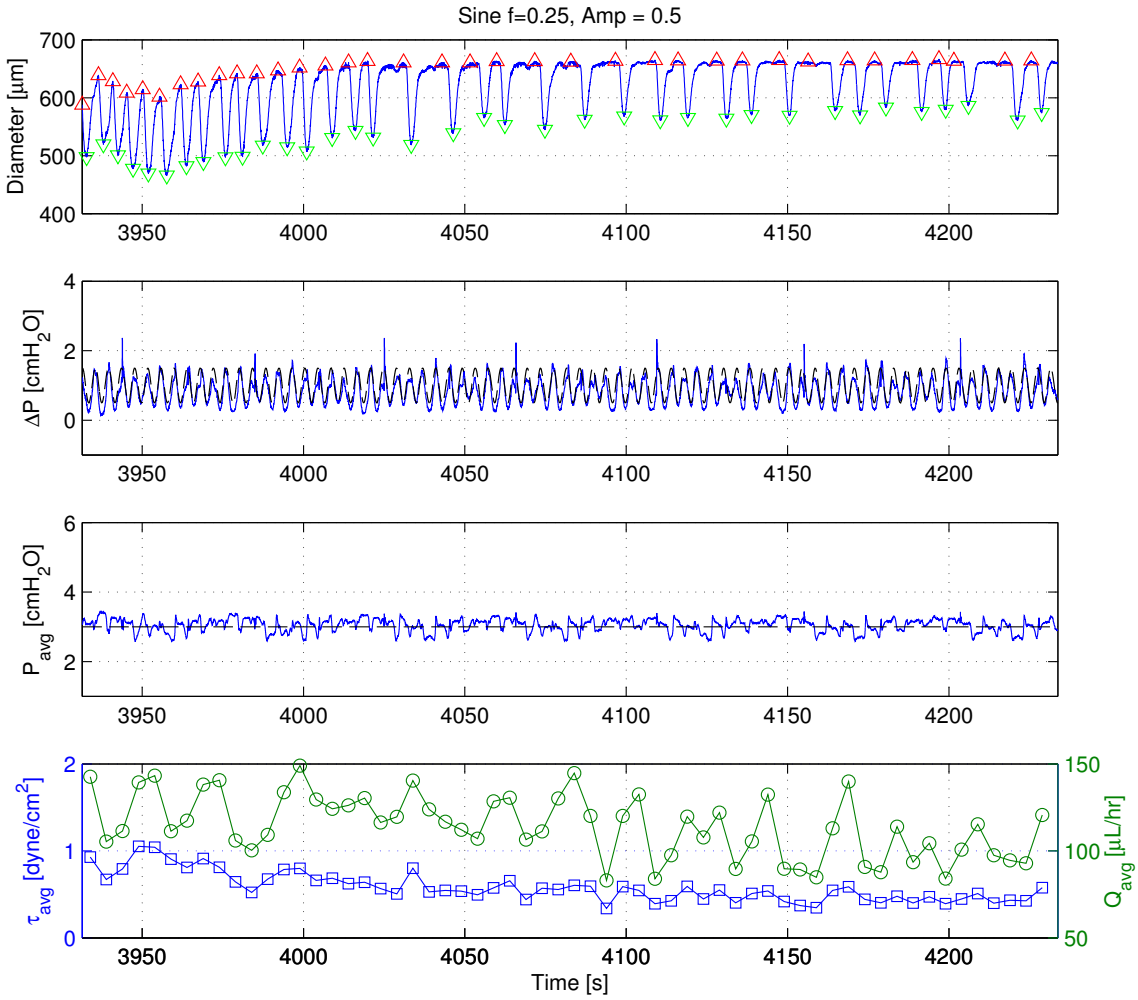


Figure C.42: 2013-08-27: 5-min Sine, $A = 0.5$ cmH_2O, $f = 0.25$ Hz.

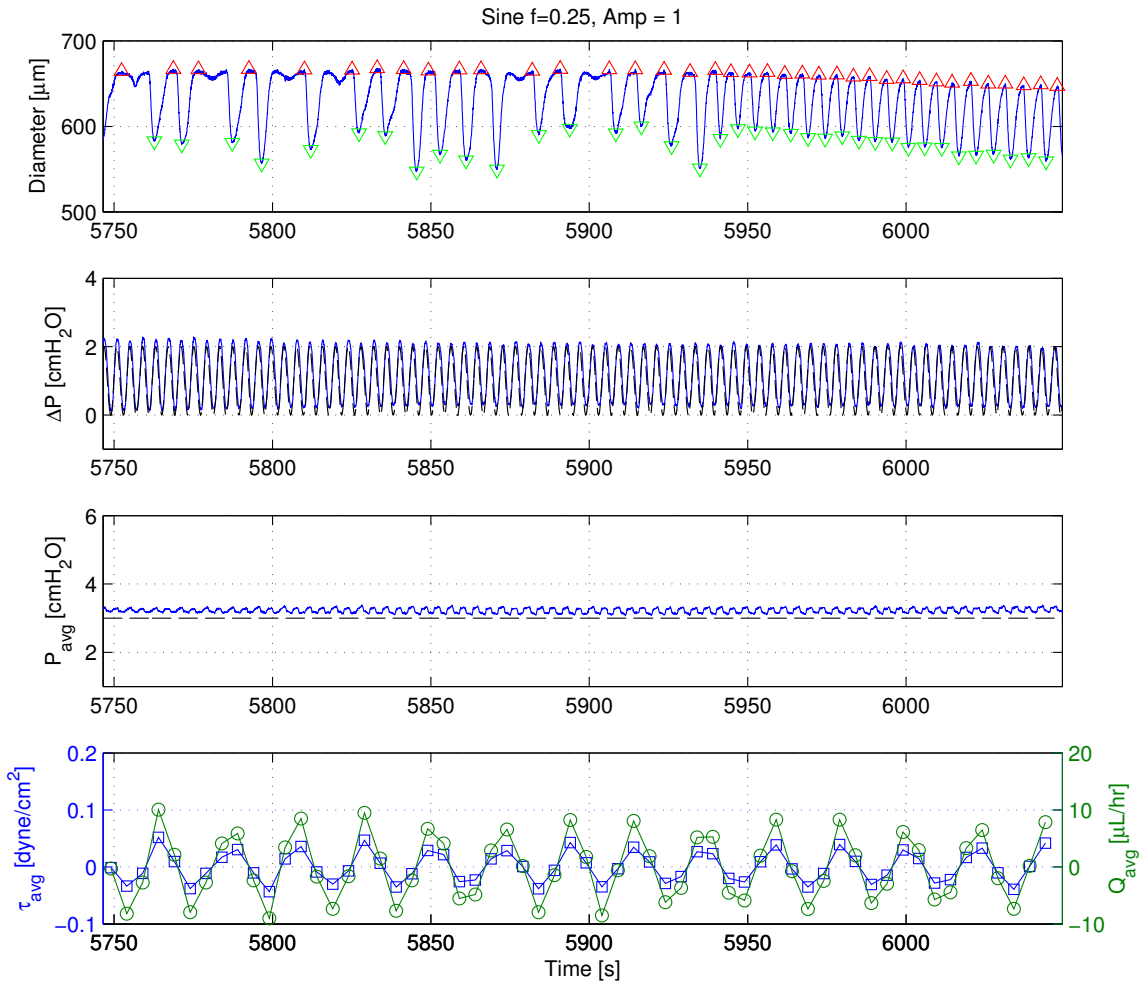


Figure C.43: 2013-08-27: 5-min Sine, $A = 1$ cmH_2O, $f = 0.25$ Hz.

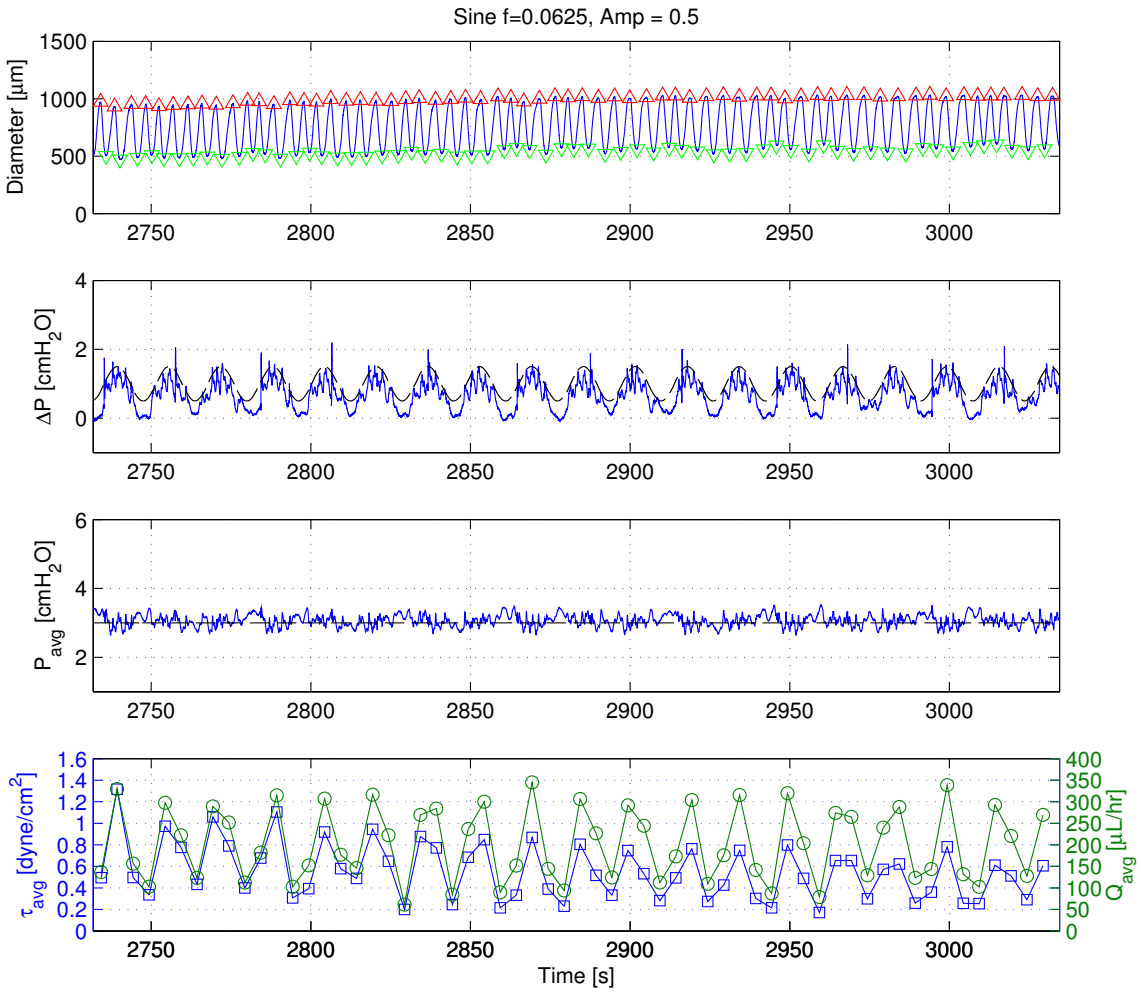


Figure C.44: 2013-08-28: 5-min Sine, $A = 0.5$ cmH$_2$O, $f = 0.0625$ Hz.

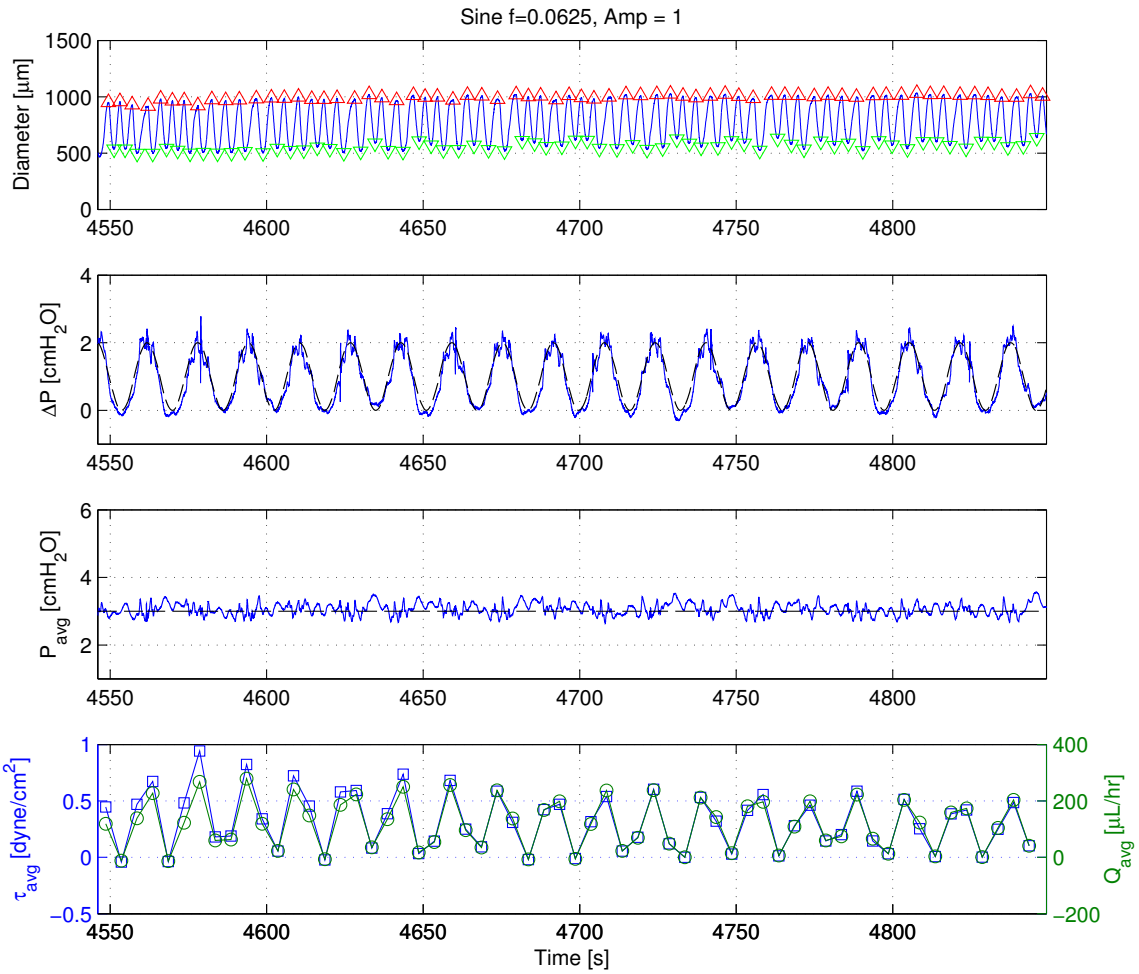


Figure C.45: 2013-08-28: 5-min Sine, $A = 1$ cmH_2O, $f = 0.0625$ Hz.

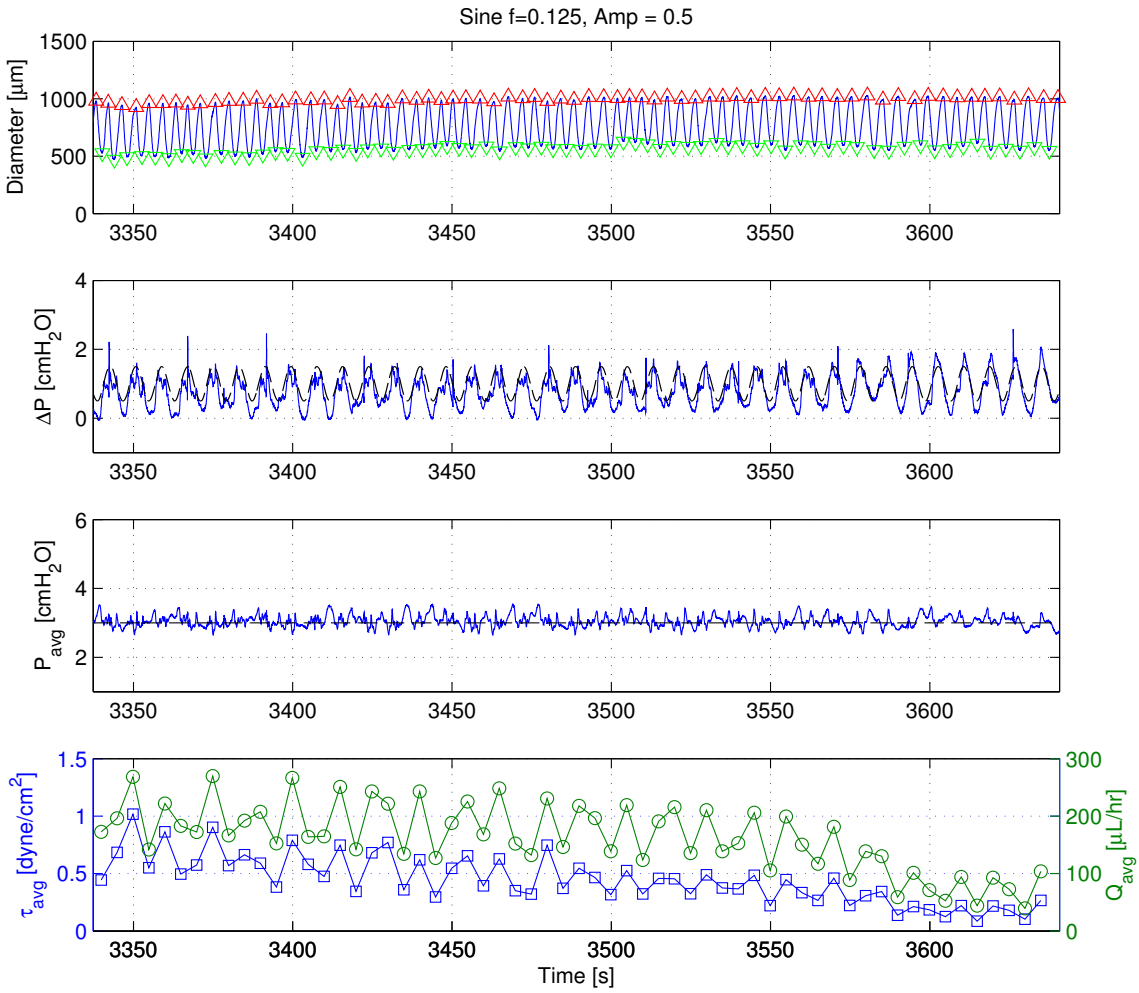


Figure C.46: 2013-08-28: 5-min Sine, $A = 0.5$ cmH_2O, $f = 0.125$ Hz.

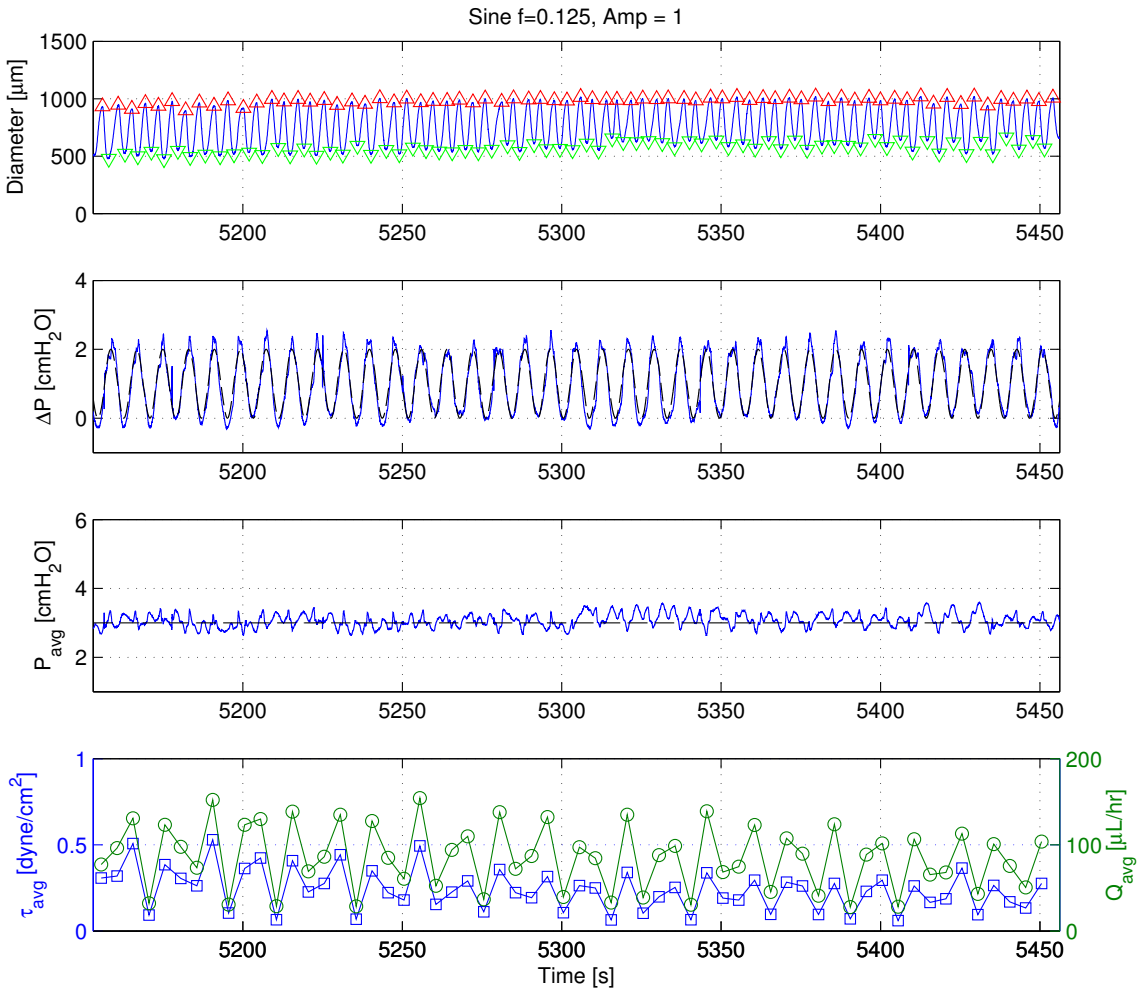


Figure C.47: 2013-08-28: 5-min Sine, $A = 1$ cmH_2O, $f = 0.125$ Hz.

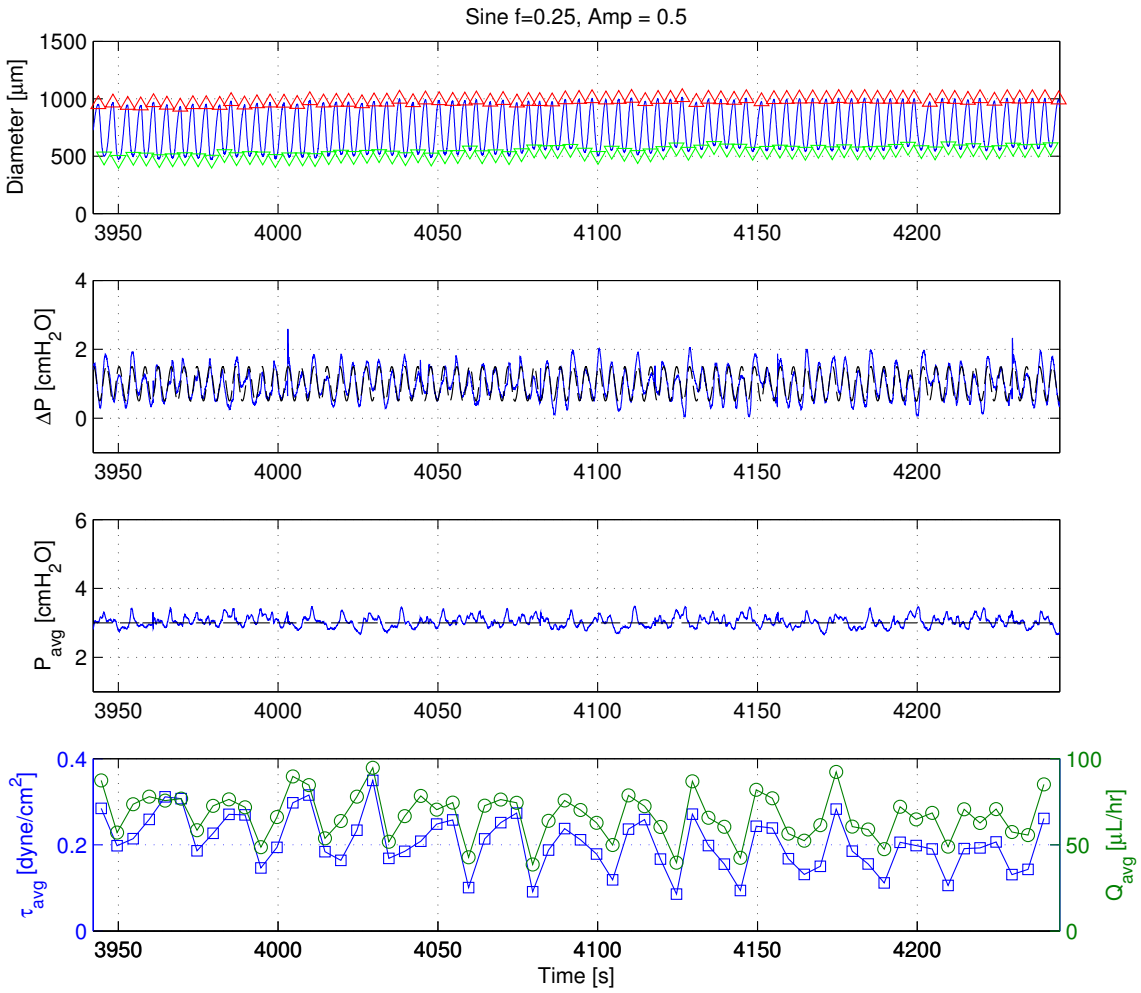


Figure C.48: 2013-08-28: 5-min Sine, $A = 0.5$ cmH$_2$O, $f = 0.25$ Hz.

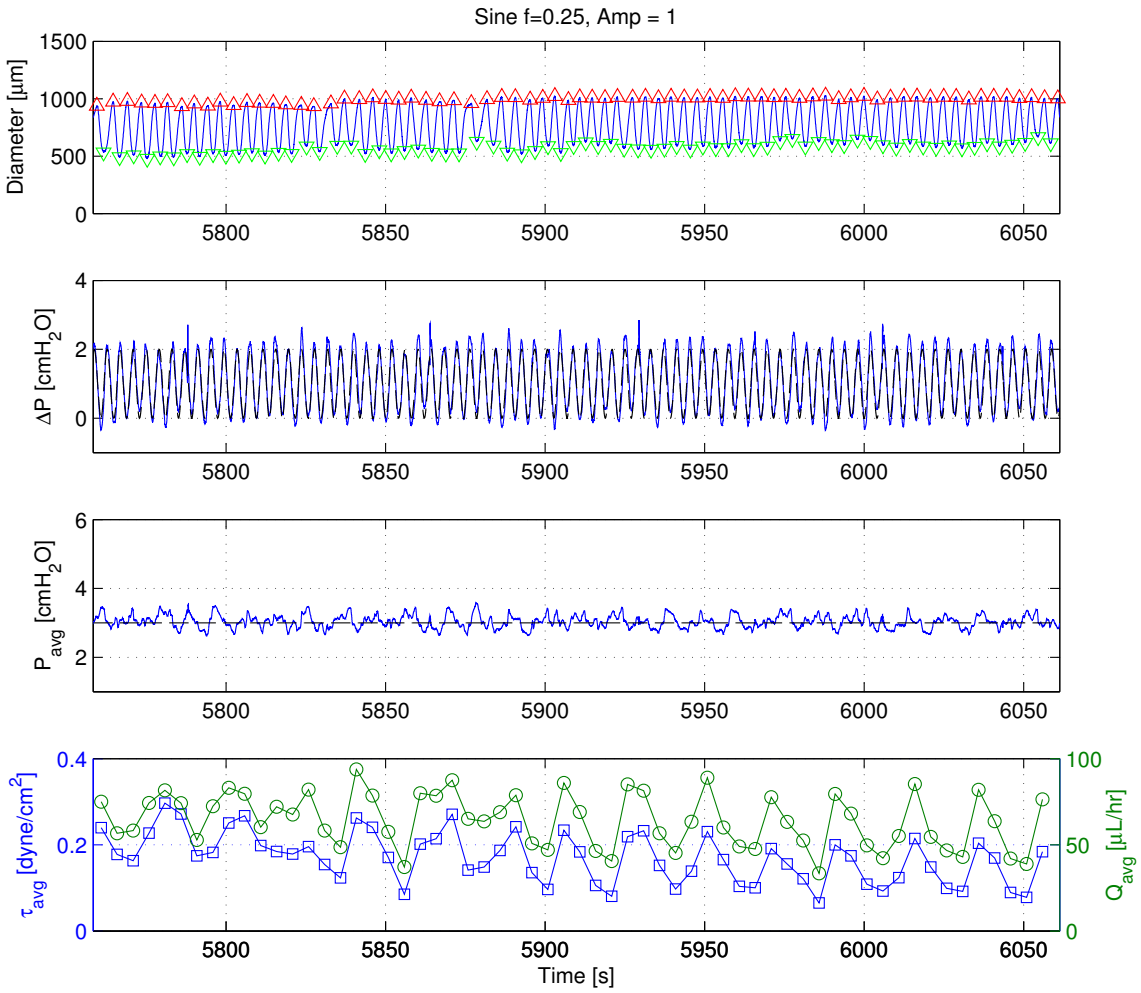


Figure C.49: 2013-08-28: 5-min Sine, $A = 1$ cmH$_2$O, $f = 0.25$ Hz.

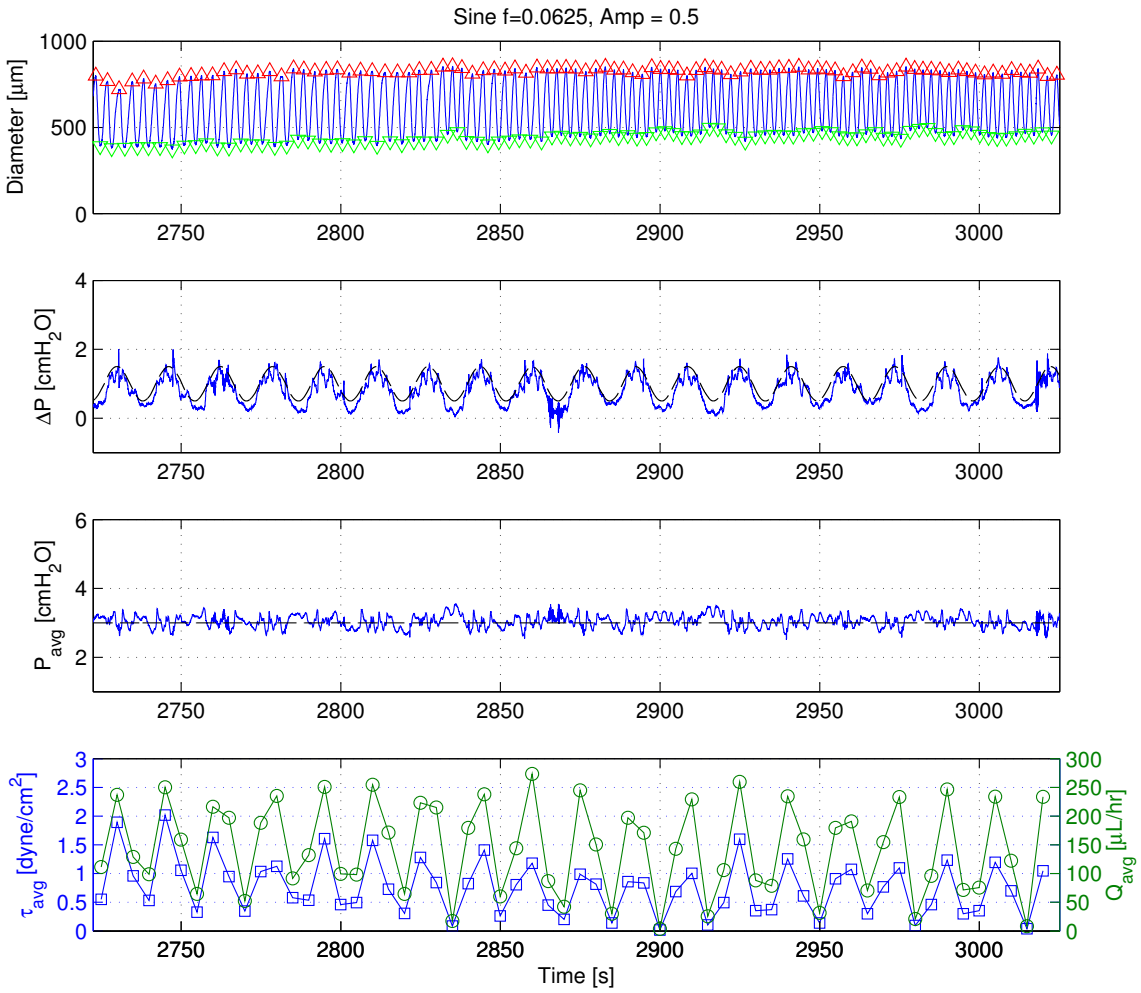


Figure C.50: 2013-08-29: 5-min Sine, $A = 0.5$ cmH$_2$O, $f = 0.0625$ Hz.

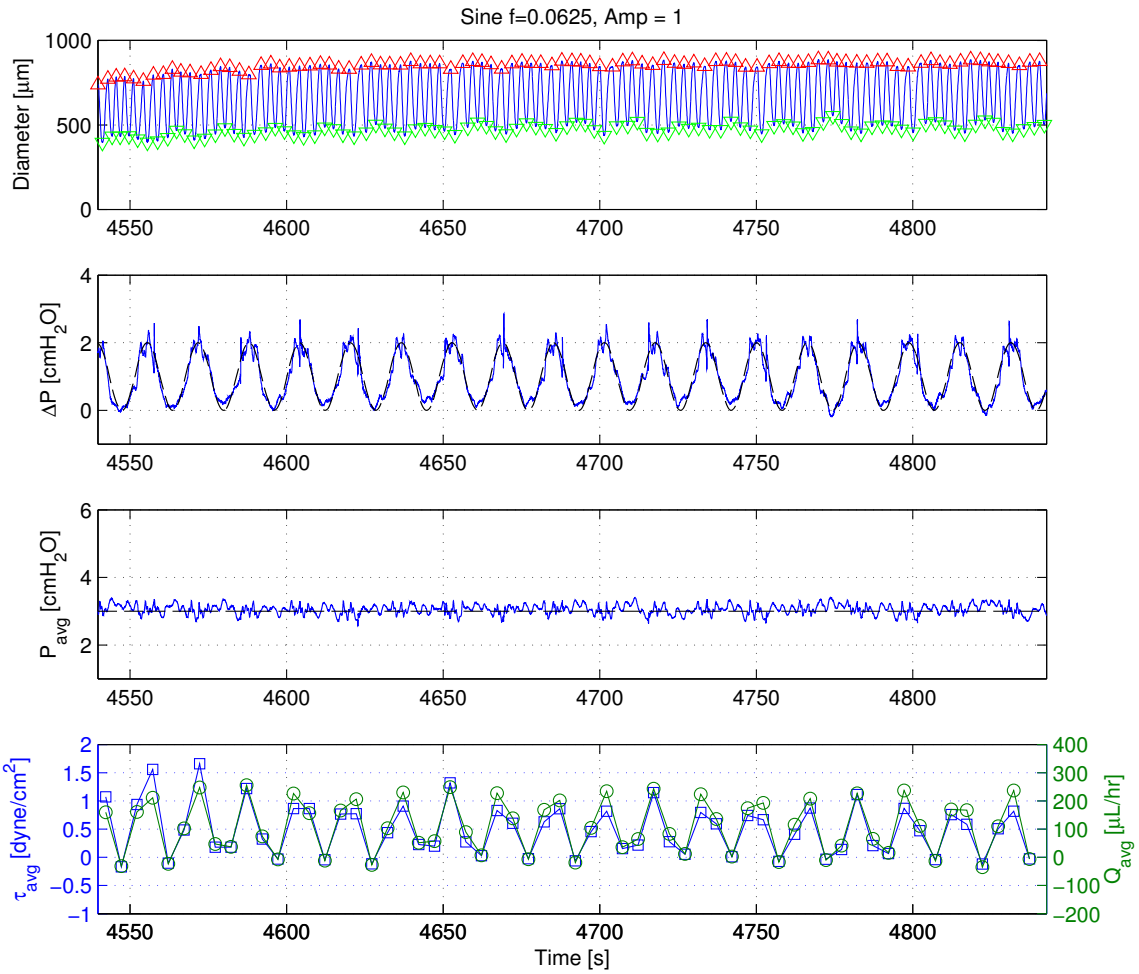


Figure C.51: 2013-08-29: 5-min Sine, $A = 1$ cmH$_2$O, $f = 0.0625$ Hz.

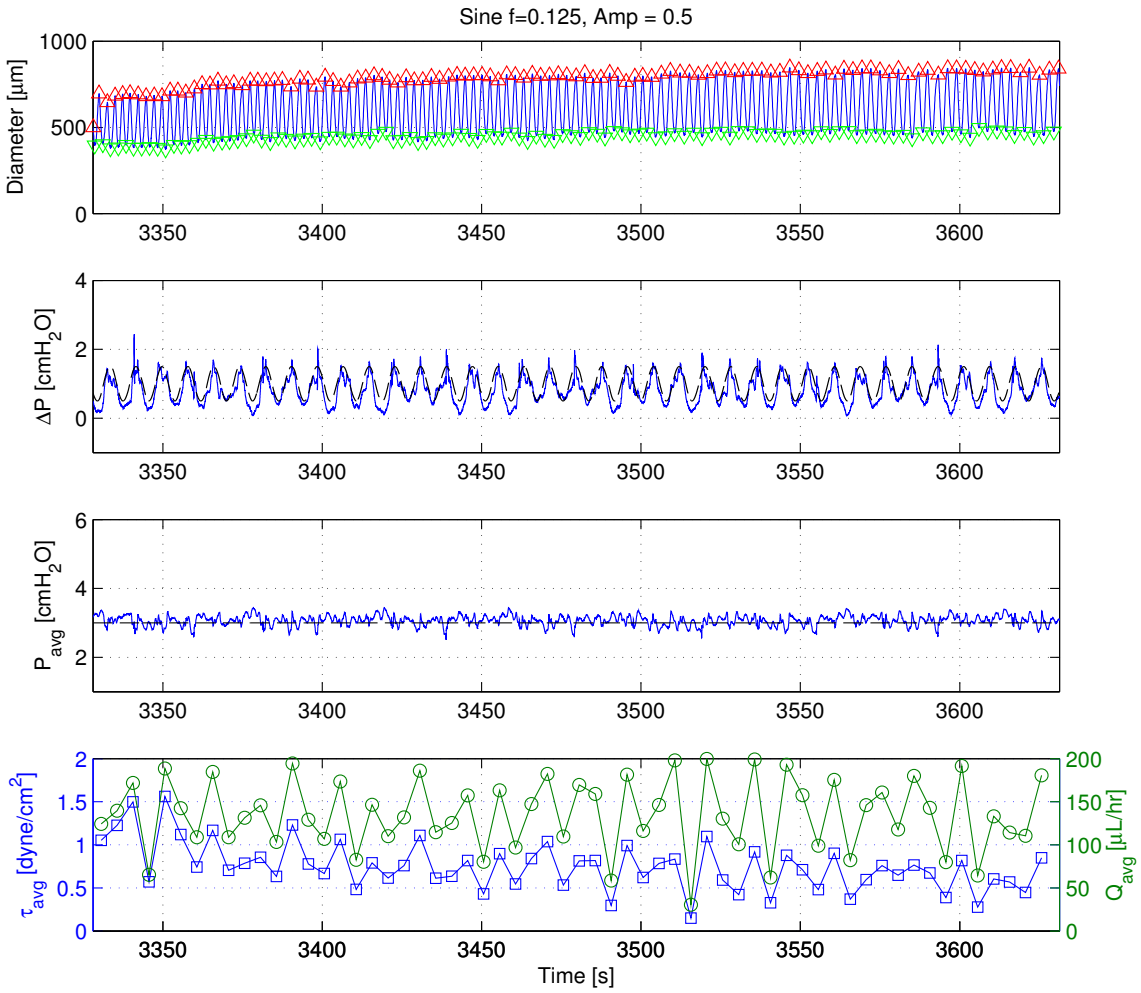


Figure C.52: 2013-08-29: 5-min Sine, $A = 0.5$ cmH$_2$O, $f = 0.125$ Hz.

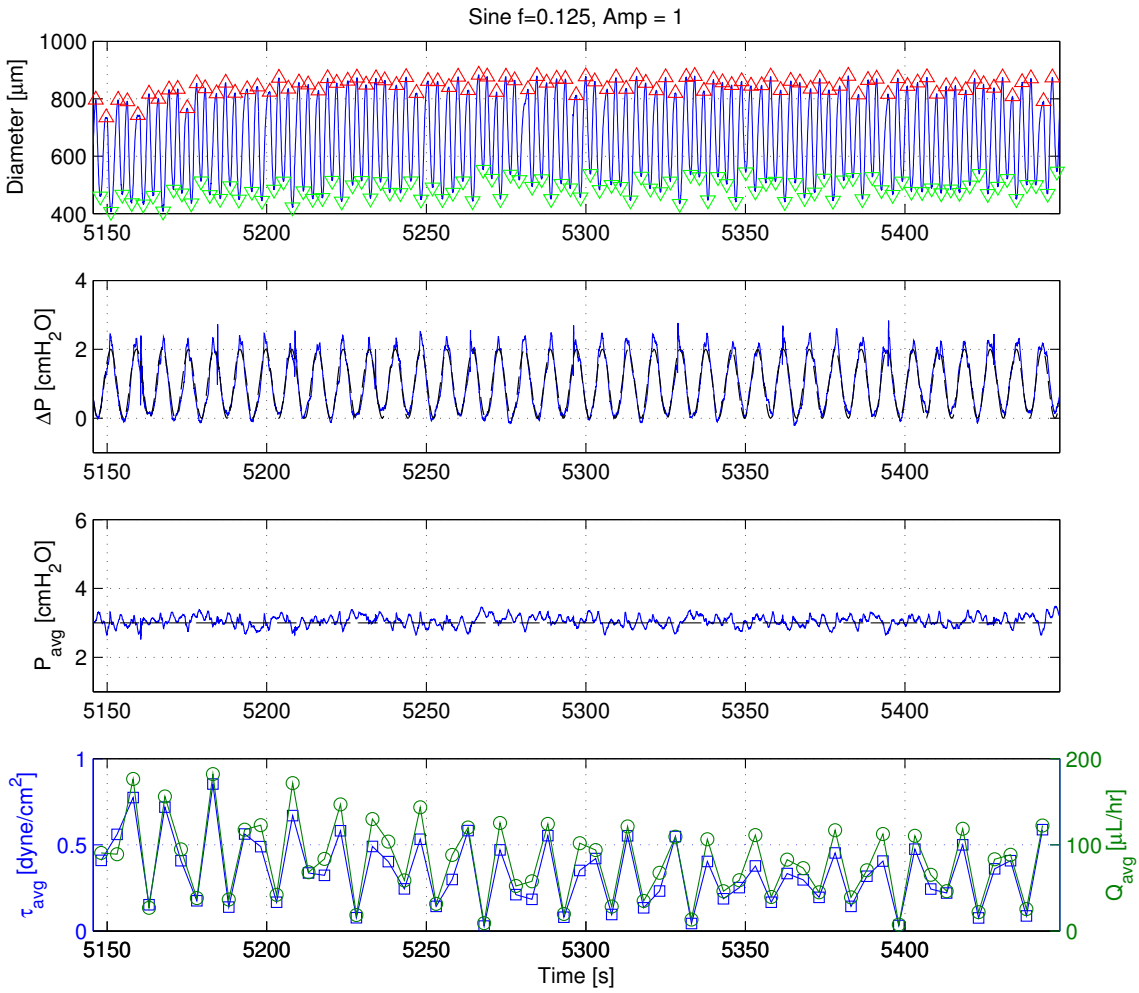


Figure C.53: 2013-08-29: 5-min Sine, $A = 1$ cmH$_2$O, $f = 0.125$ Hz.

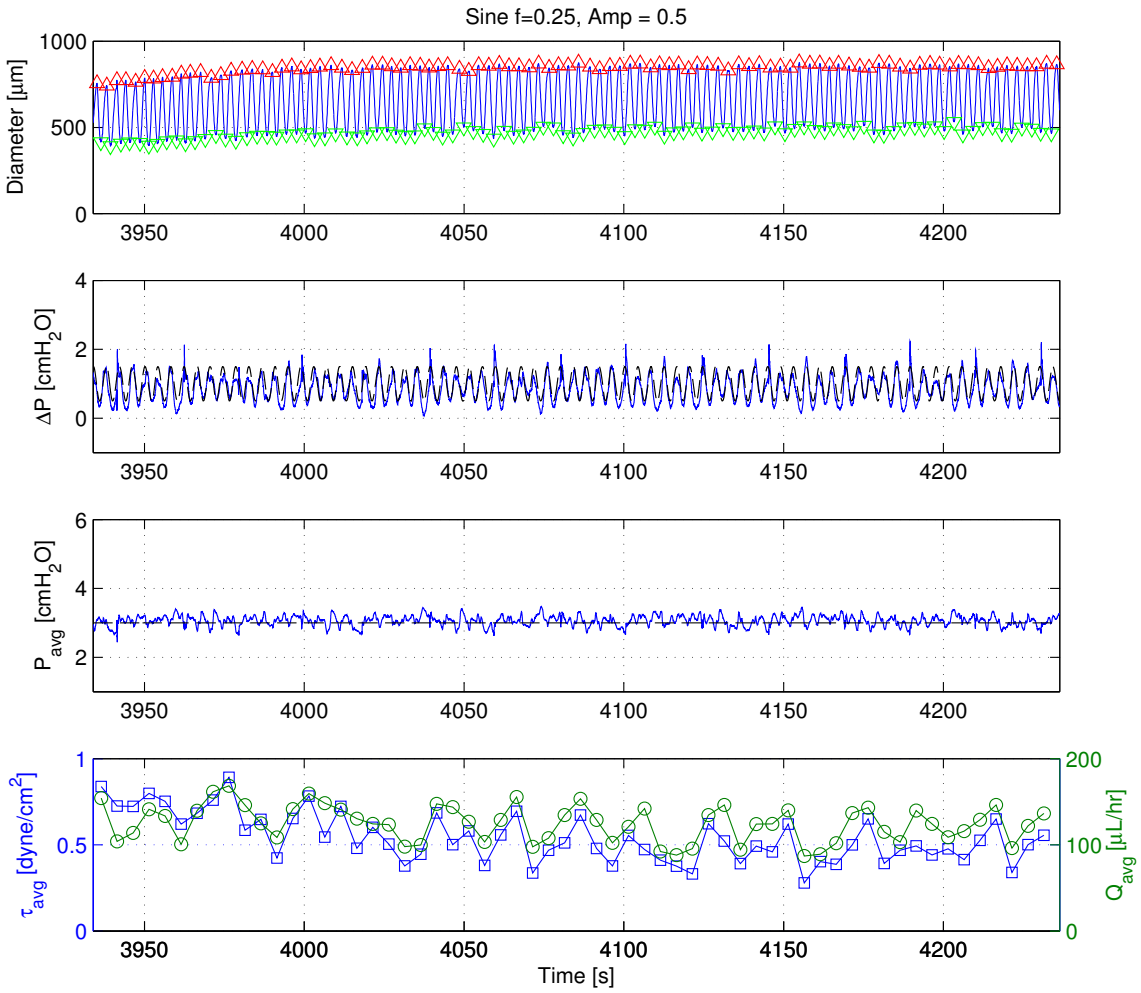


Figure C.54: 2013-08-29: 5-min Sine, $A = 0.5$ cmH$_2$O, $f = 0.25$ Hz.

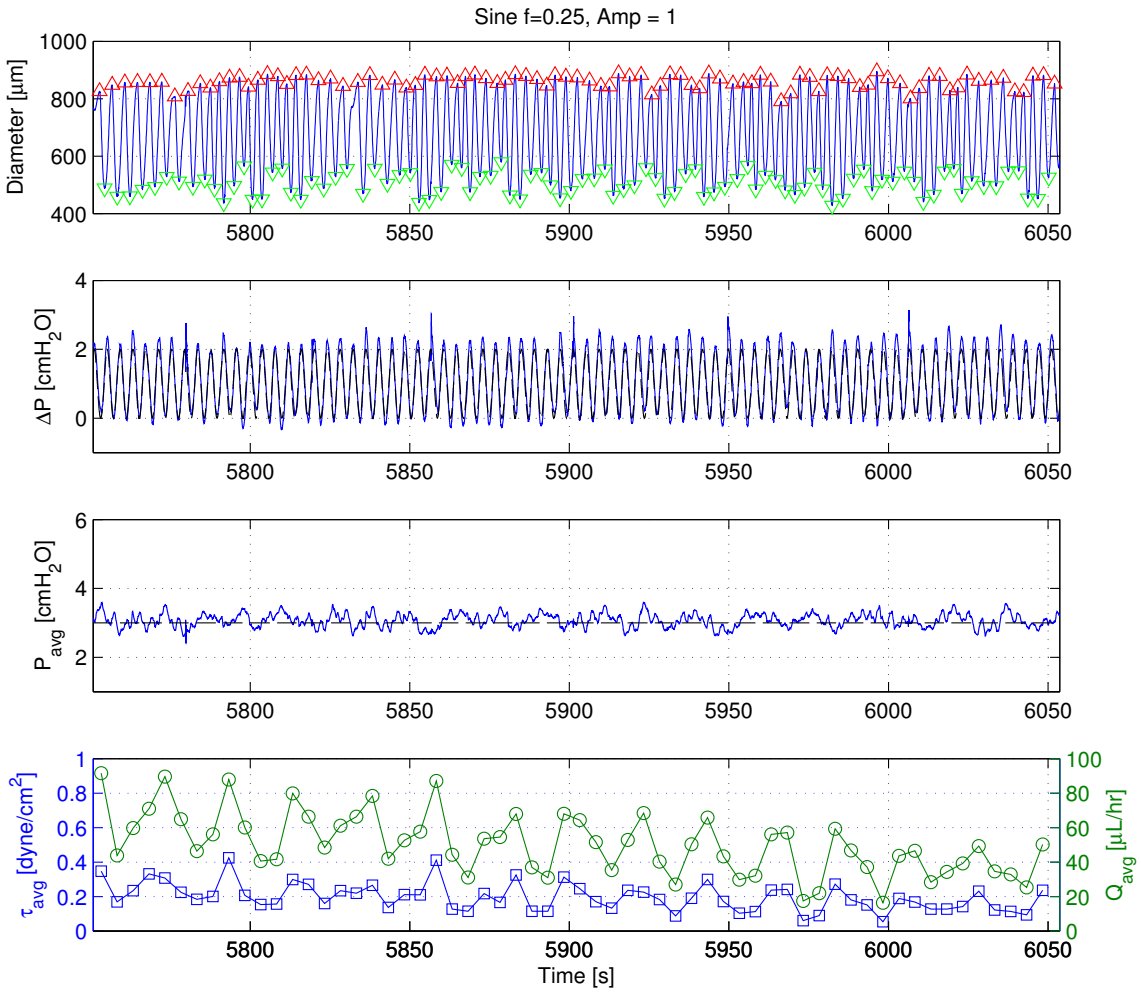


Figure C.55: 2013-08-29: 5-min Sine, $A = 1$ cmH_2O, $f = 0.25$ Hz.

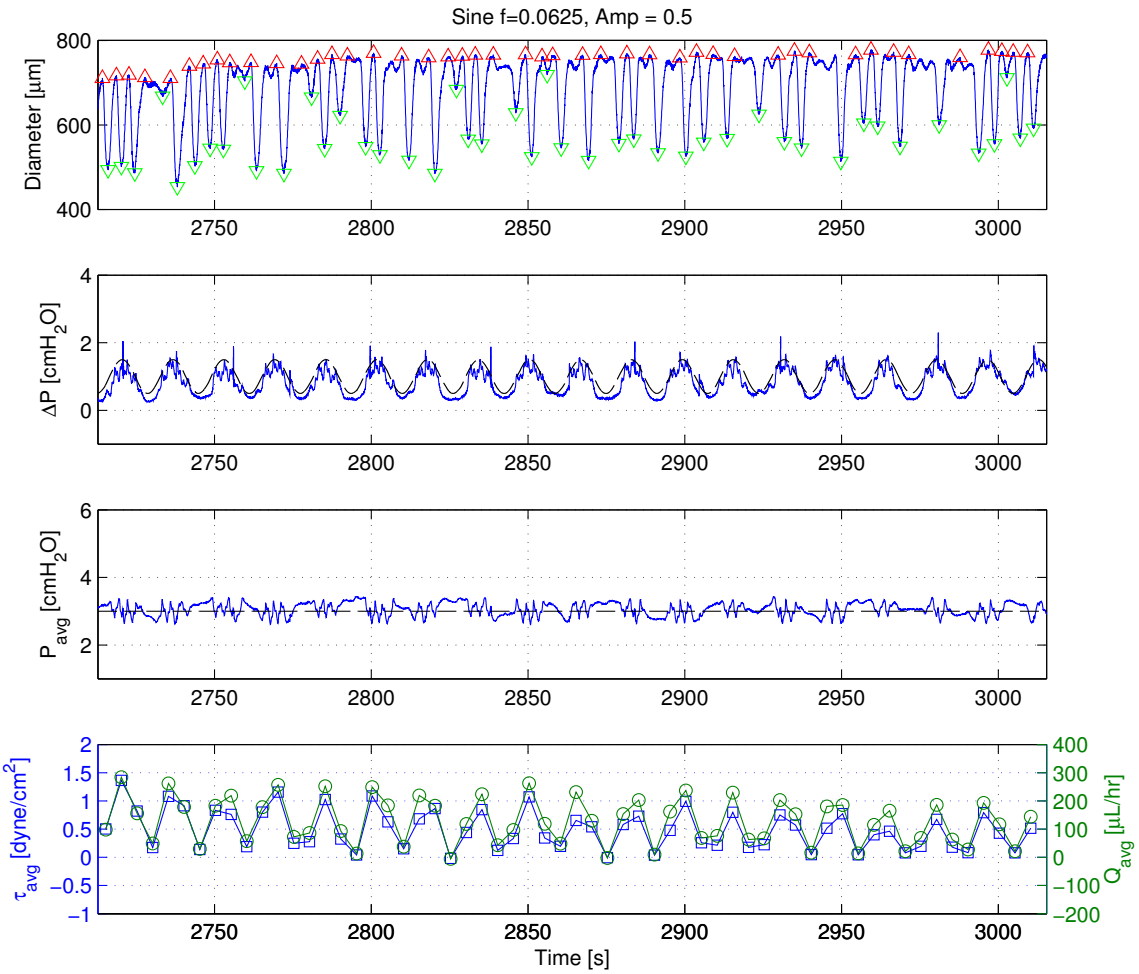


Figure C.56: 2013-09-03: 5-min Sine, $A = 0.5$ cmH_2O, $f = 0.0625$ Hz.

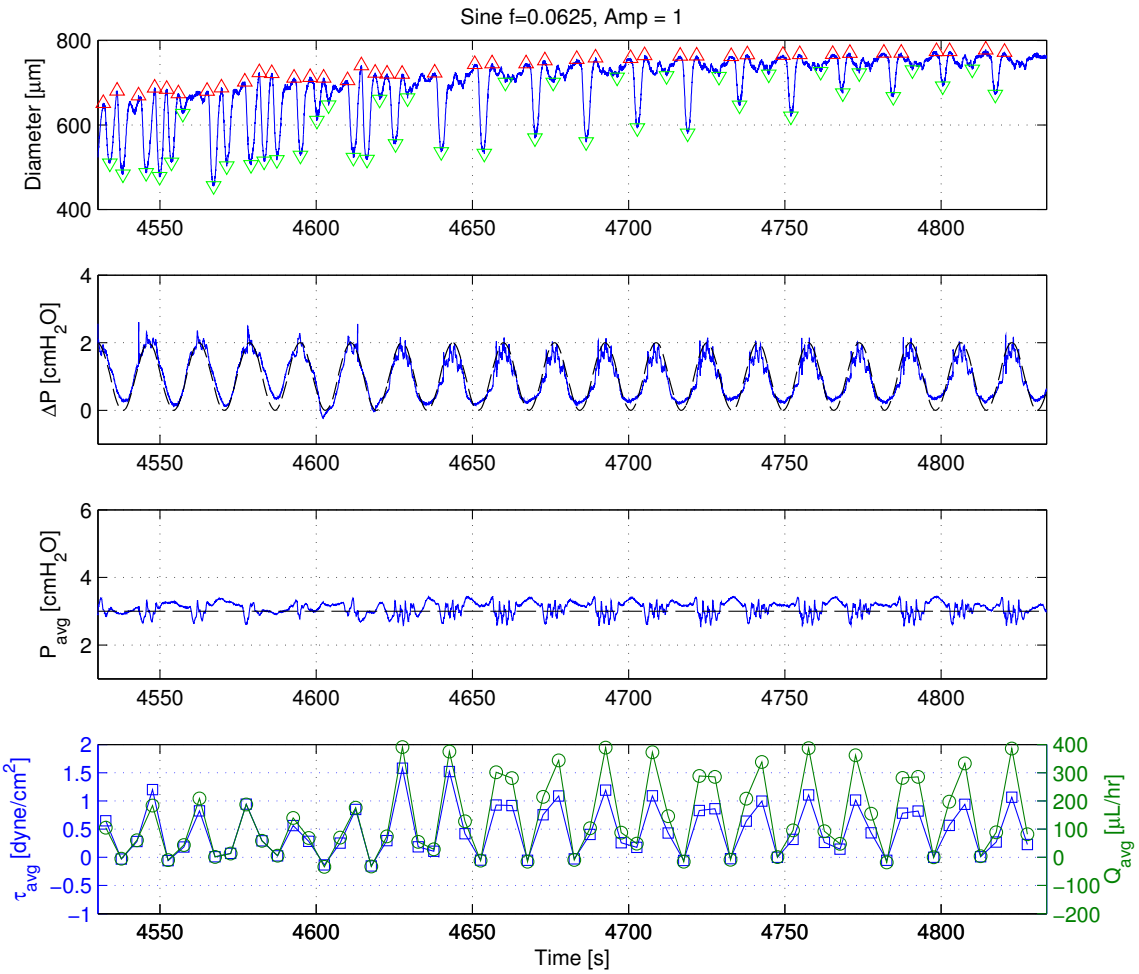


Figure C.57: 2013-09-03: 5-min Sine, $A = 1$ cmH_2O, $f = 0.0625$ Hz.

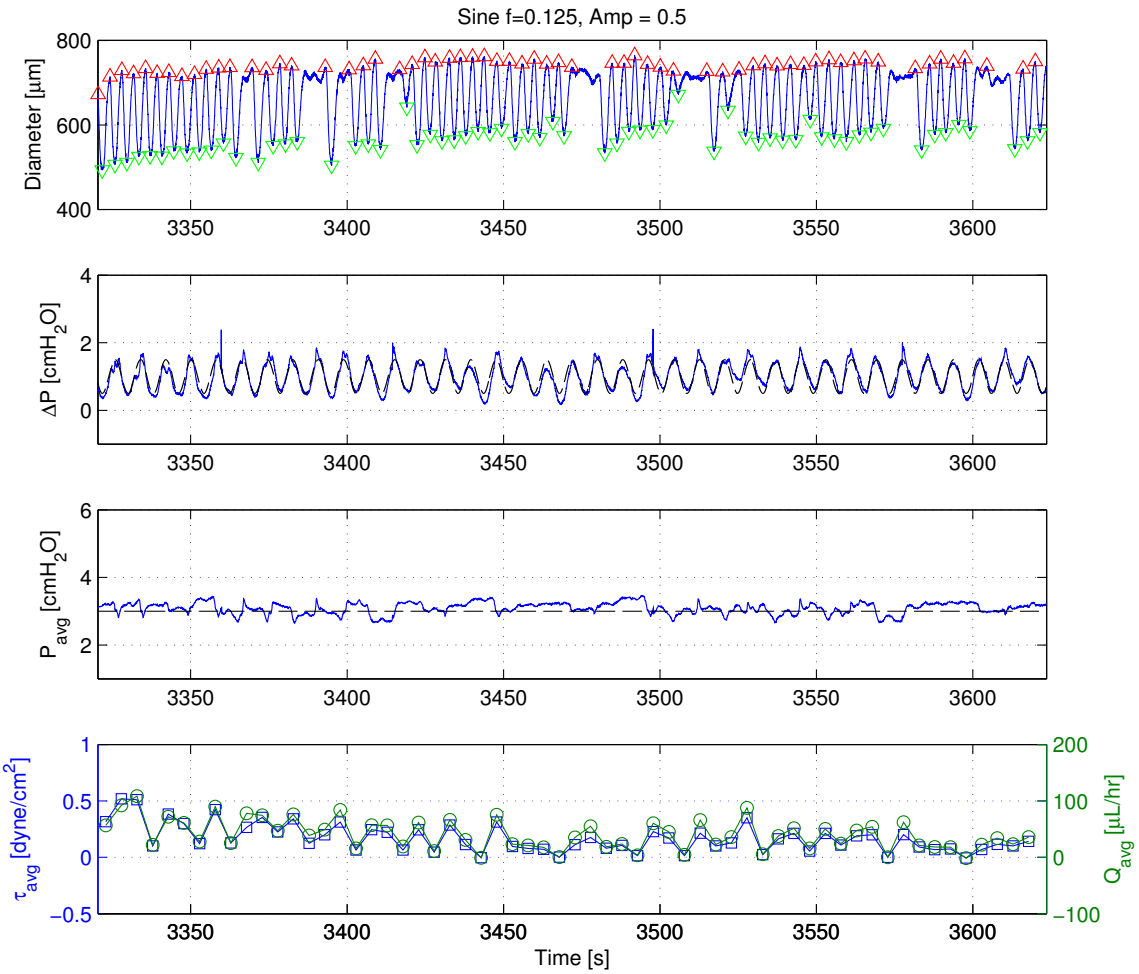


Figure C.58: 2013-09-03: 5-min Sine, $A = 0.5$ cmH_2O, $f = 0.125$ Hz.

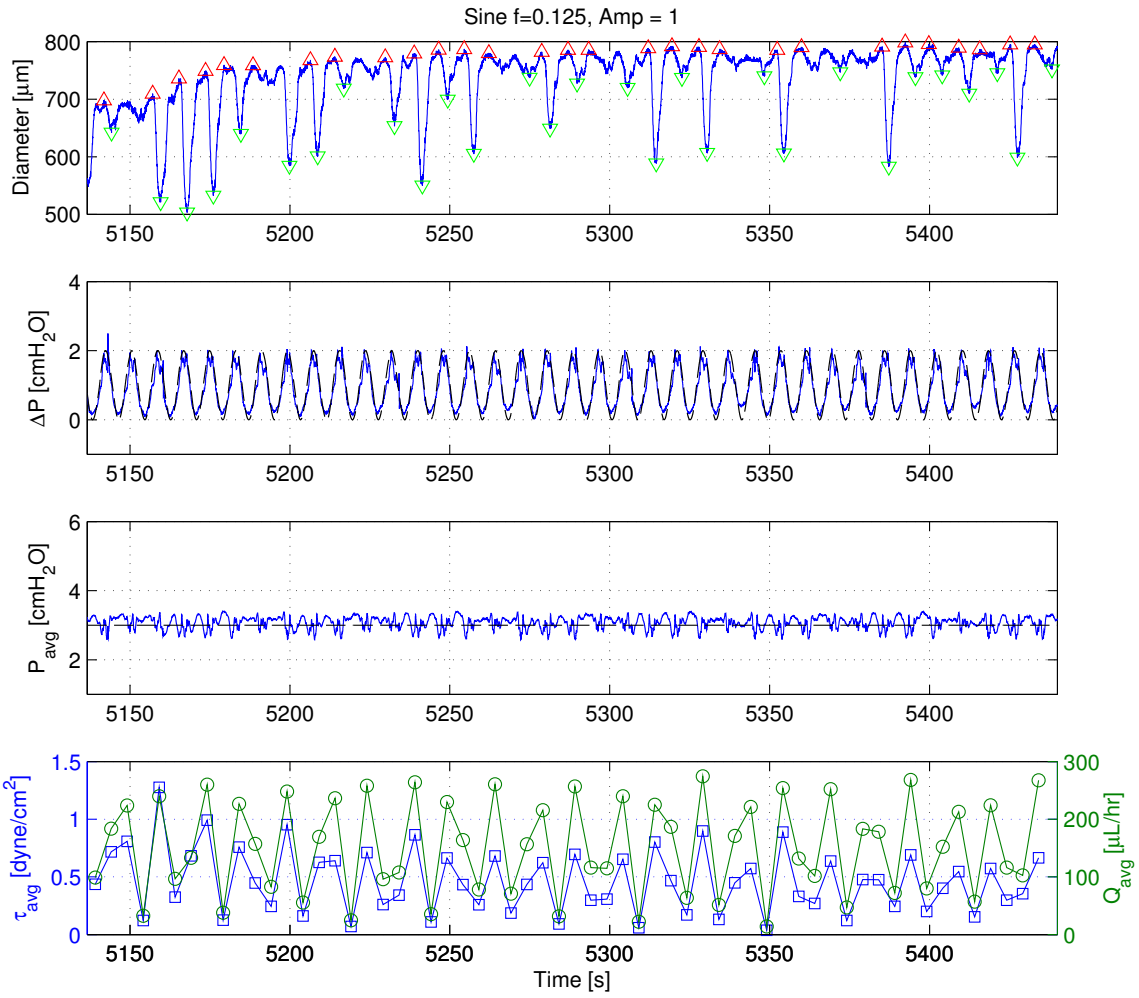


Figure C.59: 2013-09-03: 5-min Sine, $A = 1$ cmH$_2$O, $f = 0.125$ Hz.

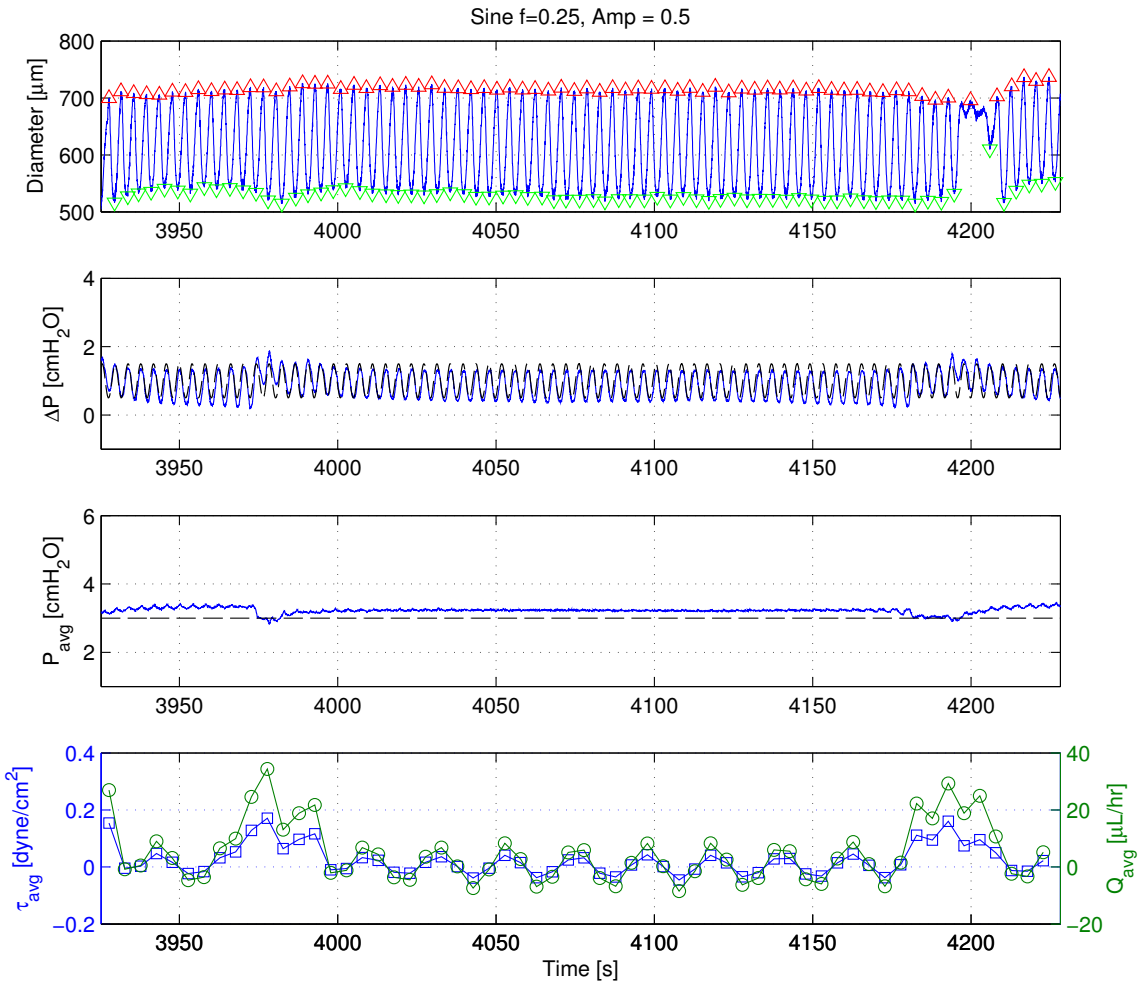


Figure C.60: 2013-09-03: 5-min Sine, $A = 0.5$ cmH_2O, $f = 0.25$ Hz.

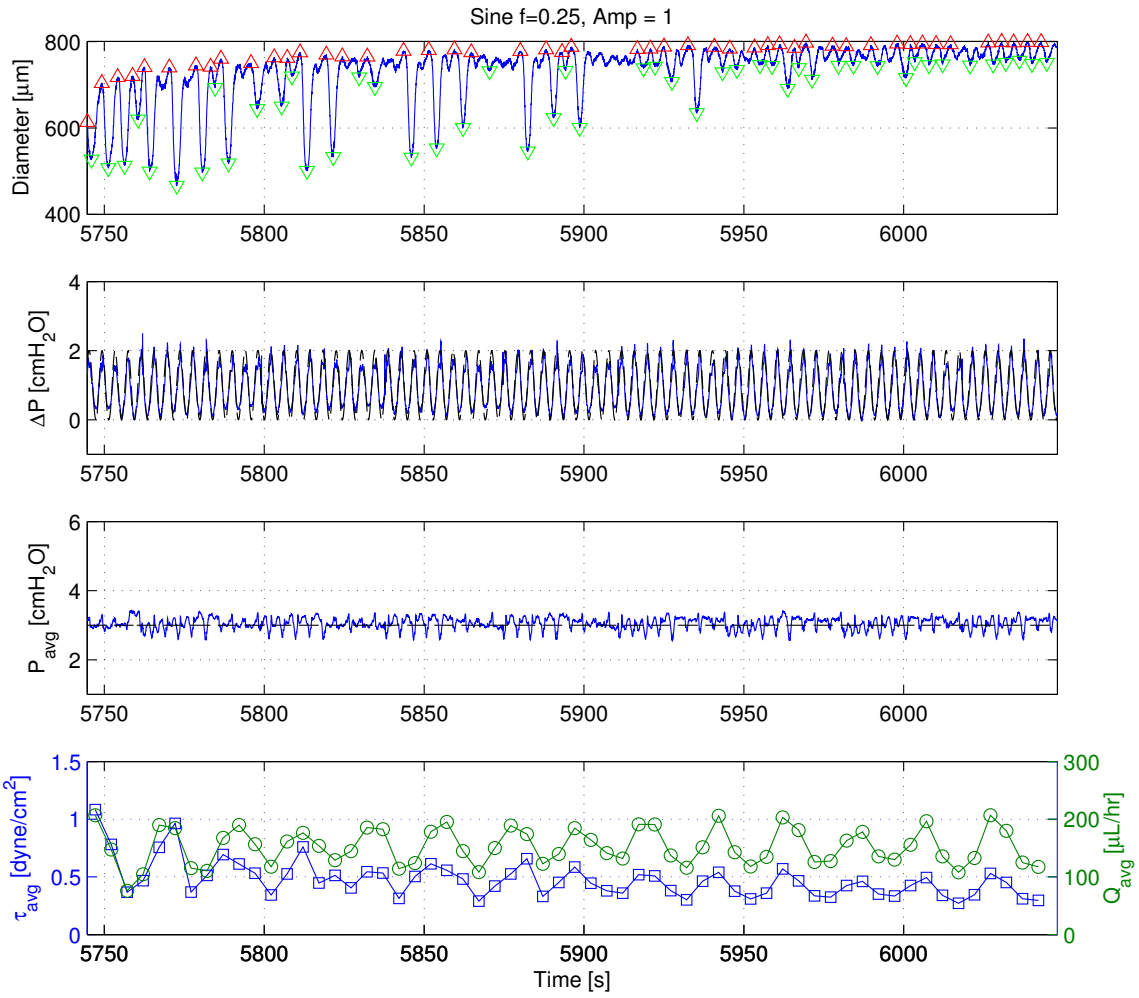


Figure C.61: 2013-09-03: 5-min Sine, $A = 1$ cmH$_2$O, $f = 0.25$ Hz.

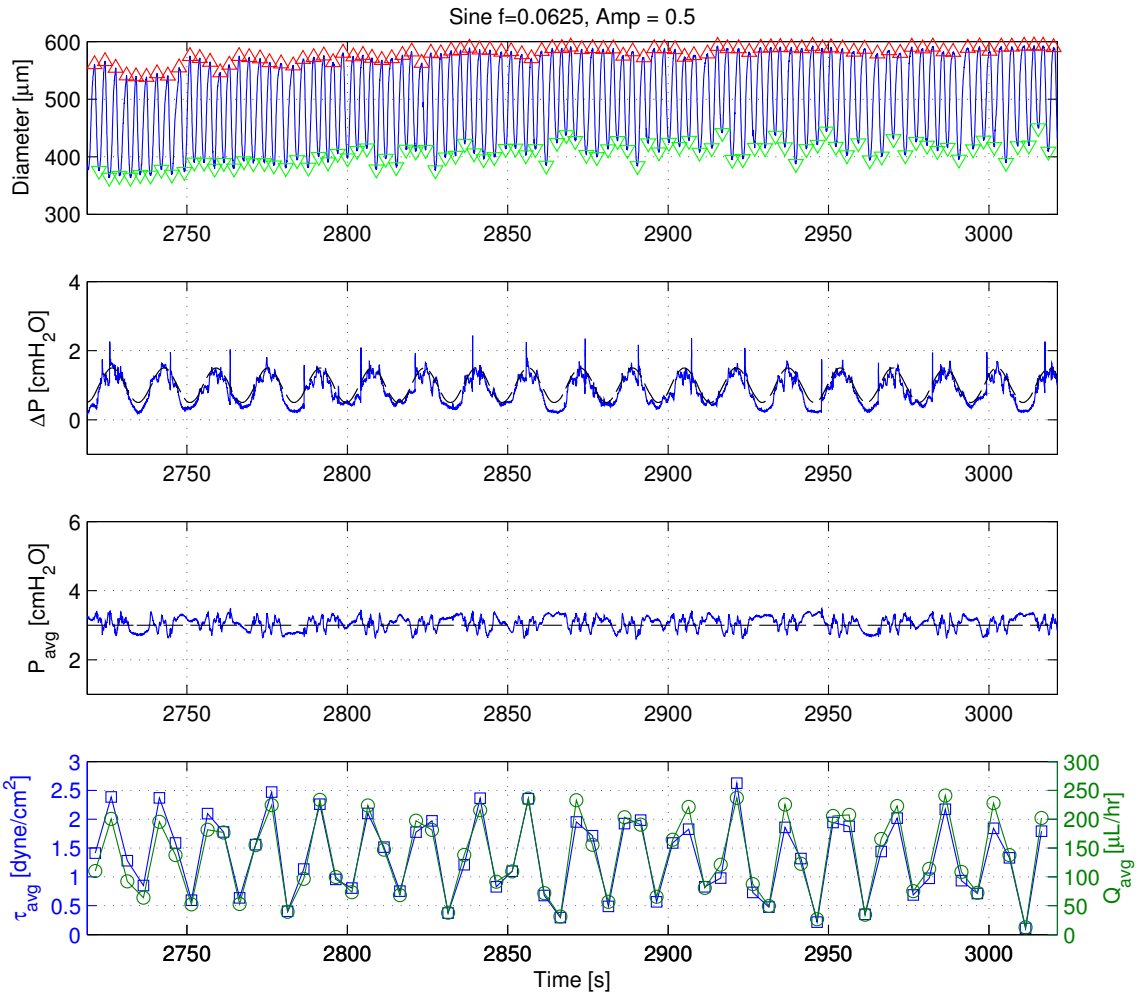


Figure C.62: 2013-09-04: 5-min Sine, $A = 0.5$ cmH_2O, $f = 0.0625$ Hz.

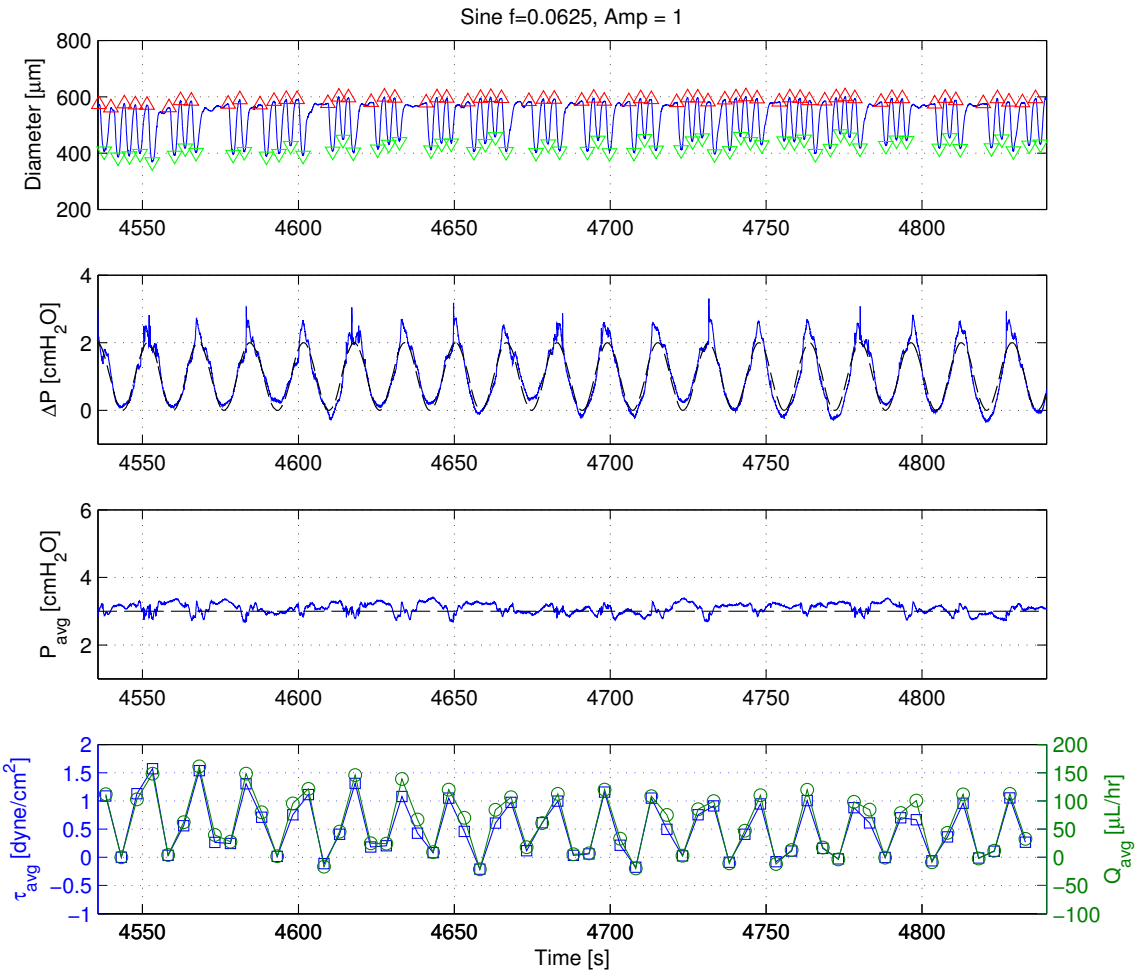


Figure C.63: 2013-09-04: 5-min Sine, $A = 1$ cmH_2O, $f = 0.0625$ Hz.

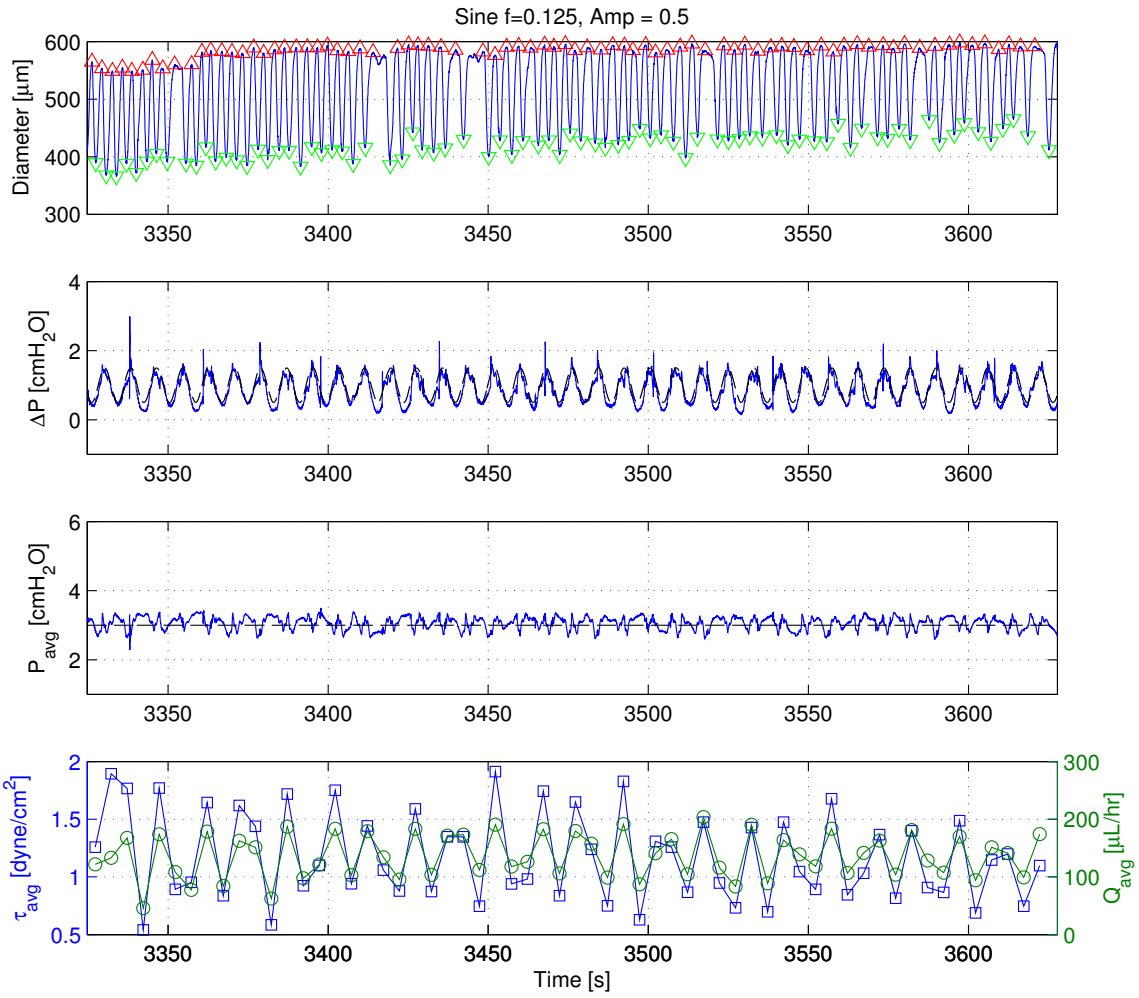


Figure C.64: 2013-09-04: 5-min Sine, $A = 0.5$ cmH$_2$O, $f = 0.125$ Hz.

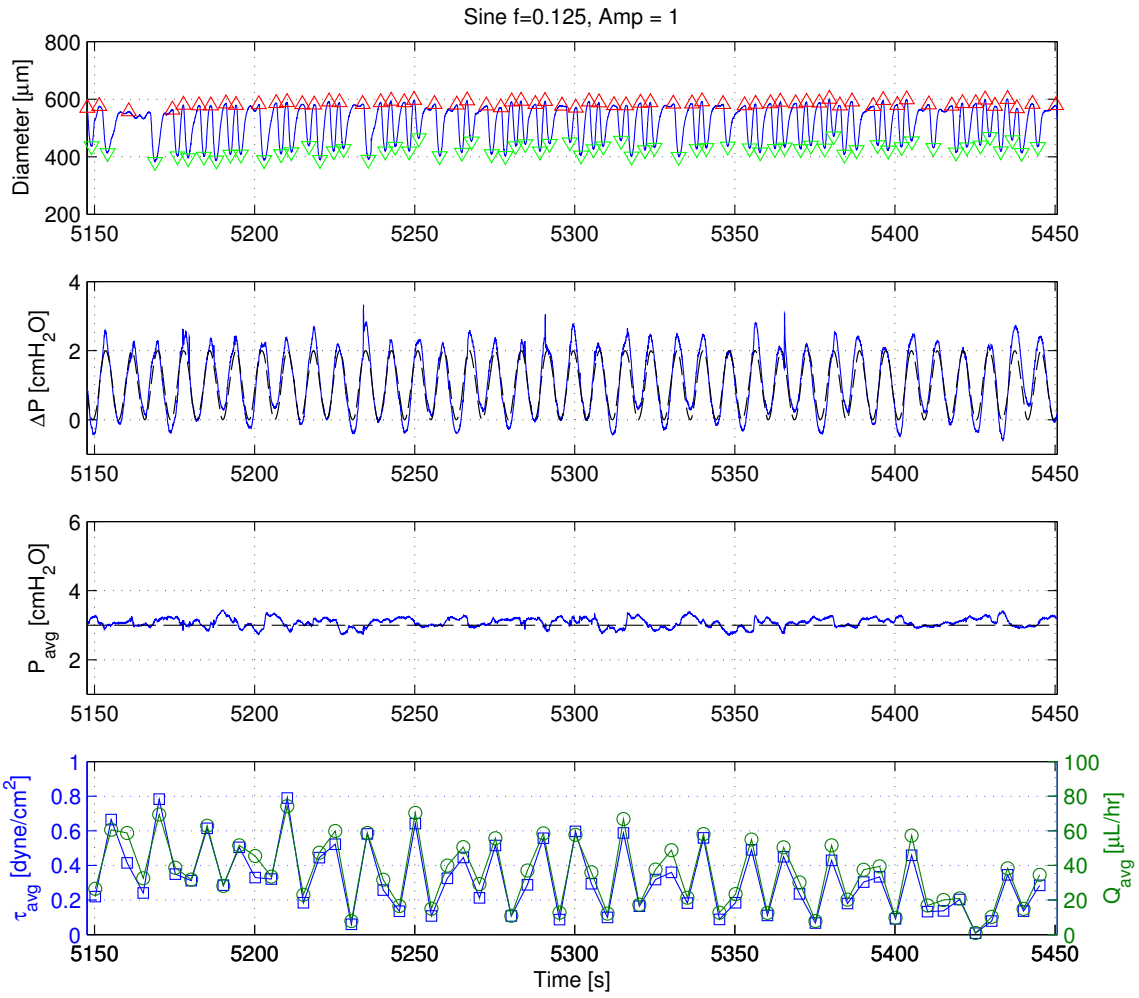


Figure C.65: 2013-09-04: 5-min Sine, $A = 1$ cmH$_2$O, $f = 0.125$ Hz.

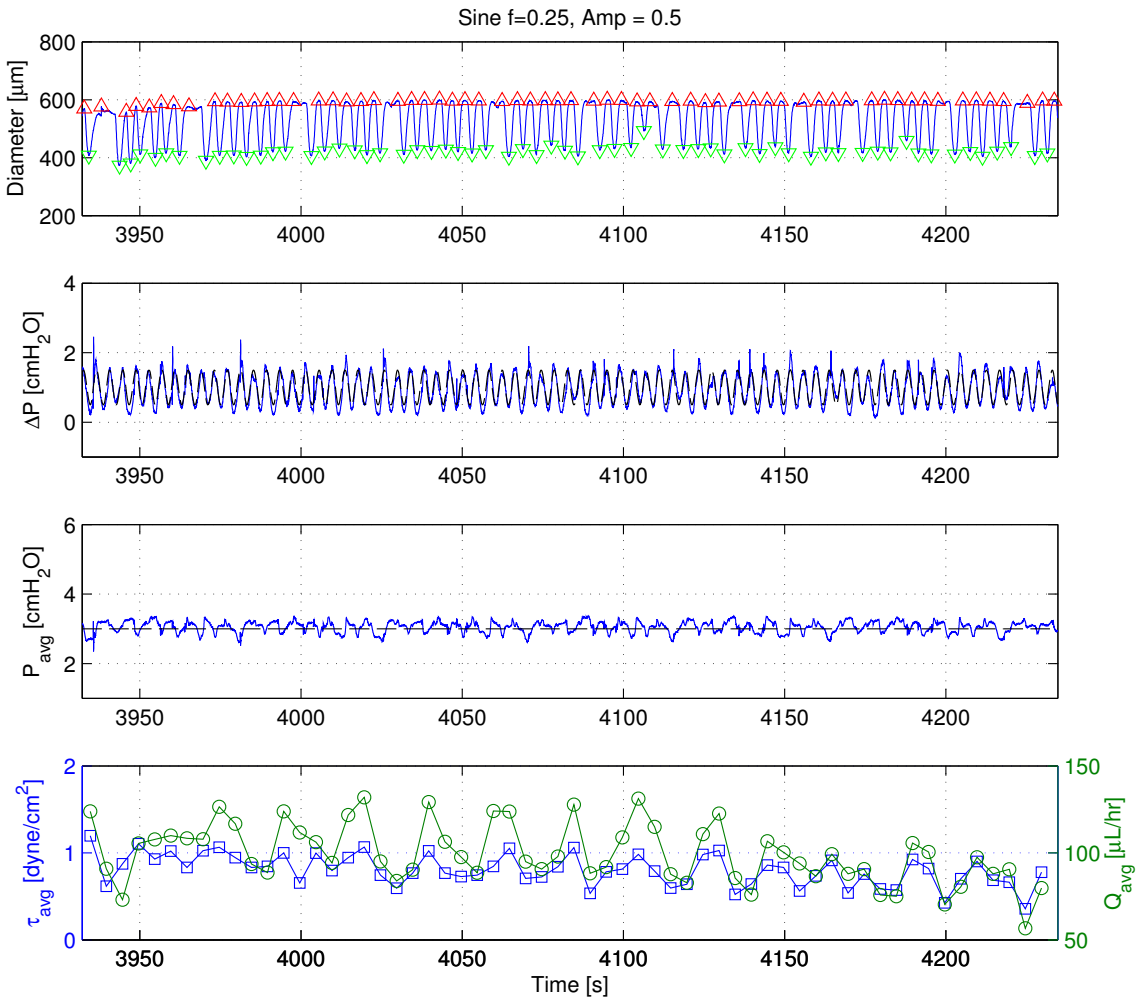


Figure C.66: 2013-09-04: 5-min Sine, $A = 0.5$ cmH_2O, $f = 0.25$ Hz.

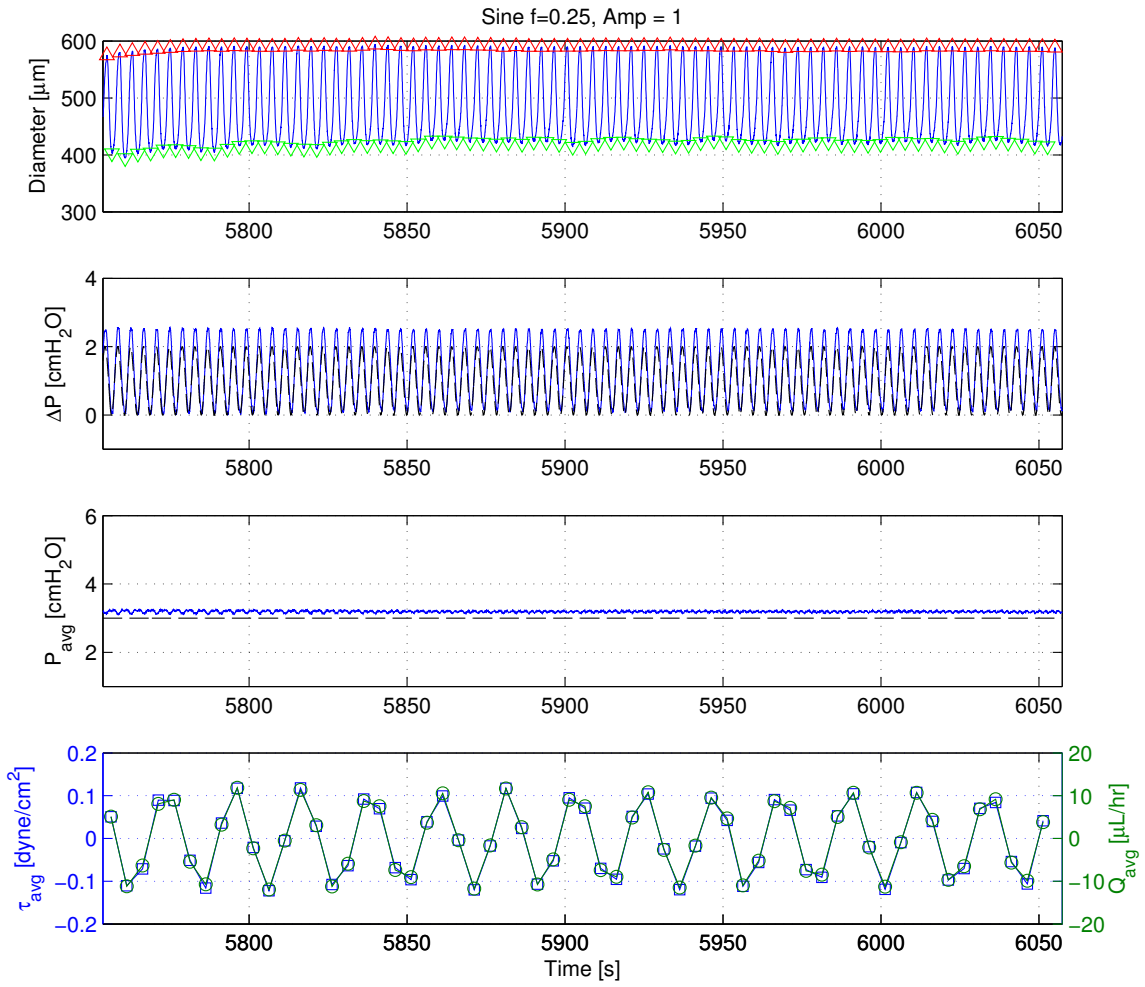


Figure C.67: 2013-09-04: 5-min Sine, $A = 1$ cmH_2O, $f = 0.25$ Hz.

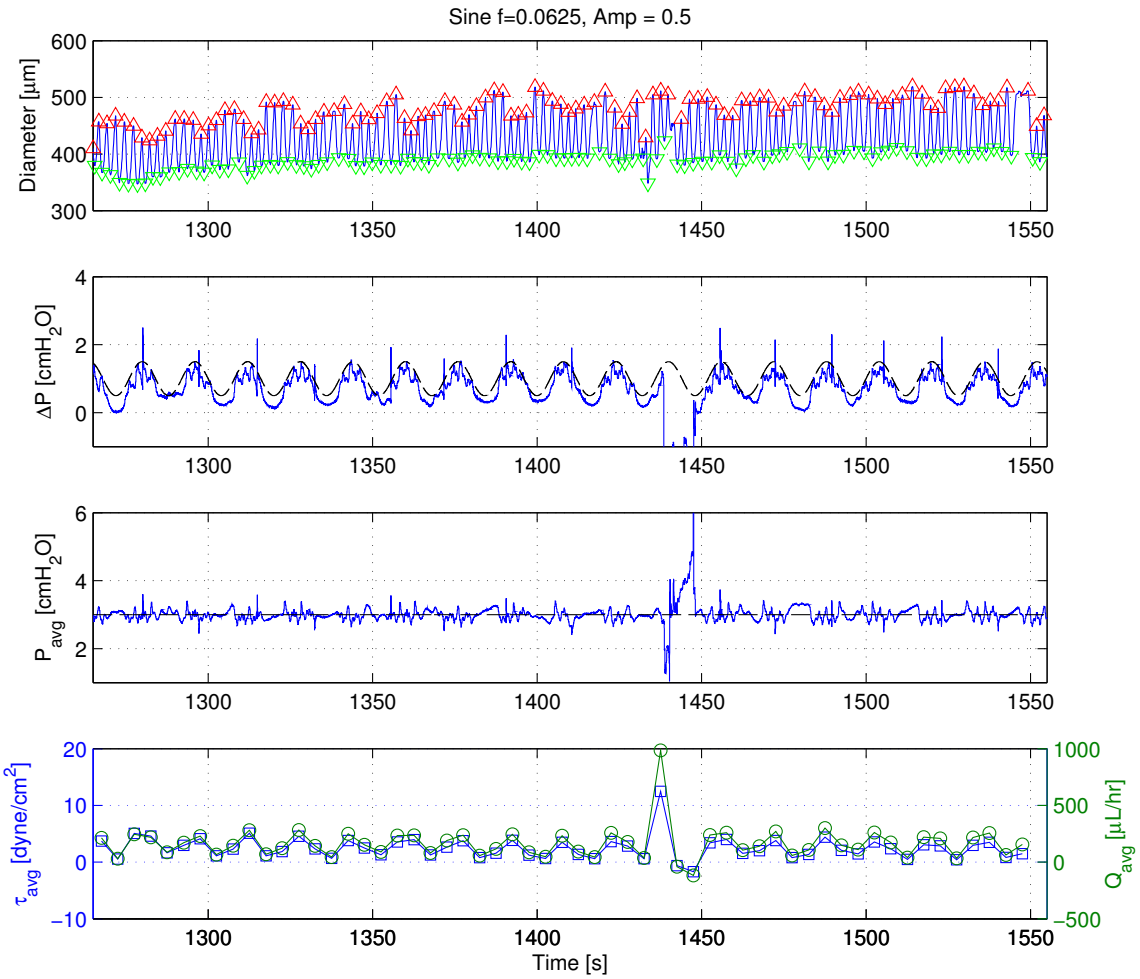


Figure C.68: 2014-03-18: 5-min Sine, $A = 0.5$ cmH_2O, $f = 0.0625$ Hz.

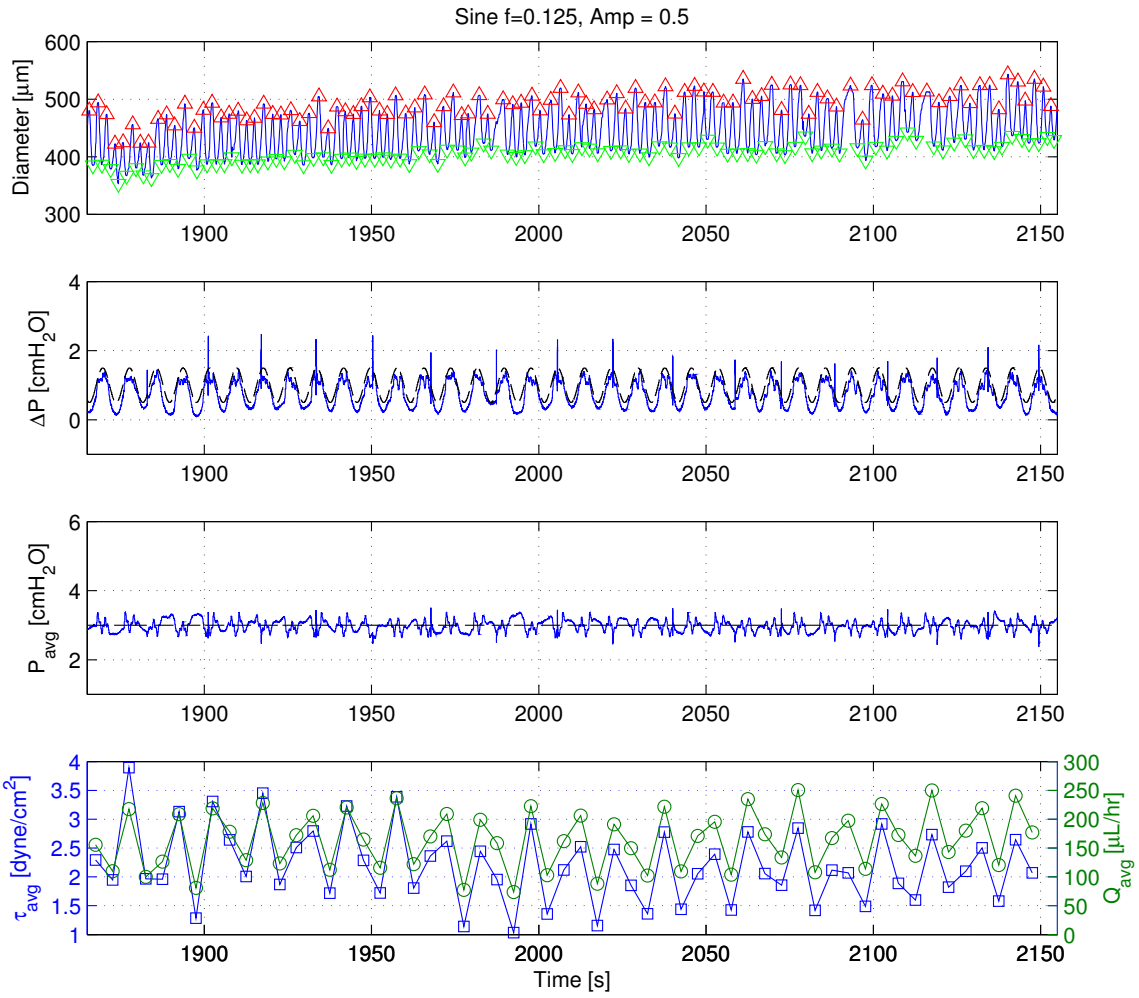


Figure C.69: 2014-03-18: 5-min Sine, $A = 0.5$ cmH_2O, $f = 0.125$ Hz.

REFERENCES

[1] BERGH, N., EKMAN, M., ULFHAMMER, E., ANDERSSON, M., KARLSSON, L., and JERN, S., "A new biomechanical perfusion system for ex vivo study of small biological intact vessels," *Annals of Biomedical Engineering*, vol. 33, pp. 1808–1818, Dec. 2005.

[2] BERTRAM, C. D., MACASKILL, C., and MOORE, J. E., "Simulation of a chain of collapsible contracting lymphangions with progressive valve closure," *Journal of Biomechanical Engineering*, vol. 133, no. 1, p. 011008, 2011.

[3] BHASKARAN, A., FERRIS, C. D., and SANDIGE, R. S., "A sampling and storage system for arbitrary biomedical waveforms.," *Biomedical Sciences Instrumentation*, vol. 29, pp. 393–399, 1993.

[4] BOHLEN, H. G., WANG, W., GASHEV, A. A., GASHEVA, O. Y., and ZAWIEJA, D. C., "Phasic contractions of rat mesenteric lymphatics increase basal and phasic nitric oxide generation in vivo," *AJP: Heart And Circulatory Physiology*, vol. 297, pp. H1319–28, Oct. 2009.

[5] CHEN, C.-Y., BERTOZZI, C., ZOU, Z., YUAN, L., LEE, J. S., LU, M., STACHELEK, S. J., SRINIVASAN, S., GUO, L., VINCENTE, A., MERICKO, P., LEVY, R. J., MAKINEN, T., OLIVER, G., and KAHN, M. L., "Blood flow reprograms lymphatic vessels to blood vessels," *The Journal of Clinical Investigation*, vol. 122, no. 6, pp. 2006–2017, 2012.

[6] CONKLIN, B. S., SUROWIEC, S. M., LIN, P. H., and CHEN, C., "A simple physiologic pulsatile perfusion system for the study of intact vascular tissue," *Medical Engineering and Physics*, vol. 22, pp. 441–449, July 2000.

[7] CONWAY, D. E., BRECKENRIDGE, M. T., HINDE, E., GRATTON, E., CHEN, C. S., and SCHWARTZ, M. A., "Fluid shear stress on endothelial cells modulates mechanical tension across VE-Cadherin and PECAM-1," *Current Biology*, vol. 23, pp. 1024–1030, June 2013.

[8] D'AUSILIO, A., "Arduino: A low-cost multipurpose lab equipment," *Behavior Research Methods*, vol. 44, pp. 305–313, Oct. 2011.

[9] DAVIS, M. J., DAVIS, A. M., LANE, M. M., KU, C. W., and GASHEV, A. A., "Rate-sensitive contractile responses of lymphatic vessels to circumferential stretch.," *The Journal of Physiology*, vol. 587, pp. 165–182, Jan. 2009.

[10] DAVIS, M. J., SCALLAN, J. P., WOLPERS, J. H., MUTHUCHAMY, M., GASHEV, A. A., and ZAWIEJA, D. C., "Intrinsic increase in lymphangion muscle contractility

in response to elevated afterload," *AJP: Heart And Circulatory Physiology*, vol. 303, pp. H795–H808, Oct. 2012.

[11] DIXON, J. B., MOORE JR, J. E., COTE, G., GASHEV, A. A., and ZAWIEJA, D. C., "Lymph flow, shear stress, and lymphocyte velocity in rat mesenteric prenodal lymphatics," *Microcirculation*, vol. 13, no. 7, pp. 597–610, 2006.

[12] DIXON, J. B., ZAWIEJA, D. C., GASHEV, A. A., and COTÉ, G. L., "Measuring microlymphatic flow using fast video microscopy," *Journal of Biomedical Optics*, vol. 10, no. 6, p. 064016, 2005.

[13] DONGAONKAR, R. M., NGUYEN, T. L., QUICK, C. M., HARDY, J., LAINE, G. A., WILSON, E., and STEWART, R. H., "Adaptation of mesenteric lymphatic vessels to prolonged changes in transmural pressure," *AJP: Heart And Circulatory Physiology*, vol. 305, pp. H203–H210, July 2013.

[14] EL-KURDI, M. S., VIPPERMAN, J. S., and VORP, D. A., "Design and subspace system identification of an ex vivo vascular perfusion system," *Journal of Biomechanical Engineering*, vol. 131, p. 041012, Apr. 2009.

[15] ESTRADA, R., GIRIDHARAN, G. A., NGUYEN, M.-D., ROUSSEL, T. J., SHAKERI, M., PARICHEHREH, V., PRABHU, S. D., and SETHU, P., "Endothelial cell culture model for replication of physiological profiles of pressure, flow, stretch, and shear stress in vitro.," *Analytical Chemistry*, vol. 83, pp. 3170–3177, Apr. 2011.

[16] FRIJNS, J. H., VAN WIJNGAARDEN, A., and PEETERS, S., "A multi-channel simultaneous data acquisition and waveform generator system designed for medical applications.," *Journal of Medical Engineering and Technology*, vol. 18, pp. 54–60, Feb. 1994.

[17] GARCÍA, C. E., PRETT, D. M., and MORARI, M., "Model predictive control: Theory and practice - a survey," *Automatica*, vol. 25, pp. 335–348, May 1989.

[18] GASHEV, A. A., DAVIS, M. J., DELP, M. D., and ZAWIEJA, D. C., "Regional variations of contractile activity in isolated rat lymphatics," *Microcirculation*, vol. 11, pp. 477–492, Sept. 2004.

[19] GASHEV, A. A., DAVIS, M. J., and ZAWIEJA, D. C., "Inhibition of the active lymph pump by flow in rat mesenteric lymphatics and thoracic duct," *Journal of Physiology*, vol. 540, pp. 1023–1037, Apr. 2002.

[20] GASHEVA, O. Y., ZAWIEJA, D. C., and GASHEV, A. A., "Contraction-initiated NO-dependent lymphatic relaxation: a self-regulatory mechanism in rat thoracic duct," *The Journal of Physiology*, vol. 575, pp. 821–832, Sept. 2006.

[21] GEWILLIG, M., "The Fontan Circulation," *Heart (British Cardiac Society)*, vol. 91, pp. 839–846, June 2005.

[22] GLEASON, R. L., GRAY, S. P., WILSON, E., and HUMPHREY, J. D., "A multi-axial computer-controlled organ culture and biomechanical device for mouse carotid arteries," *Journal of Biomechanical Engineering*, vol. 126, pp. 787–795, Dec. 2004.

[23] GRETENER, S. B., LAUCHLI, S., LEU, A. J., KOPPENSTEINER, R., and FRANZECK, U., "Effect of venous and lymphatic congestion on lymph capillary pressure of the skin in healthy volunteers and patients with lymph edema," *Journal of Vascular Research*, vol. 37, no. 1, pp. 61–67, 2000.

[24] HARGENS, A. R. and ZWEIFACH, B. W., "Contractile stimuli in collecting lymph vessels.," *American Journal of Physiology*, vol. 233, pp. H57–65, July 1977.

[25] HELMLINGER, G., GEIGER, R. V., SCHRECK, S., and NEREM, R. M., "Effects of pulsatile flow on cultured vascular endothelial cell morphology.," *Journal of Biomechanical Engineering*, vol. 113, pp. 123–131, May 1991.

[26] HOLDSWORTH, D. W., RICKEY, D. W., DRANGOVA, M., MILLER, D. J., and FENSTER, A., "Computer-controlled positive displacement pump for physiological flow simulation," *Medical and Biological Engineering and Computing*, vol. 29, pp. 565–570, Nov. 1991.

[27] HOR, P. J., ZHU, Z. Q., HOWE, D., and REES-JONES, J., "Minimization of cogging force in a linear permanent magnet motor," *IEEE Transactions on Magnetics*, vol. 34, pp. 3544–3547, Sept. 1998.

[28] HUMPHREY, J. D., "Vascular adaptation and mechanical homeostasis at tissue, cellular, and sub-cellular levels," *Cell Biochemistry and Biophysics*, vol. 50, no. 2, pp. 53–78, 2008.

[29] IVES, C. L., ESKIN, S. G., and MCINTIRE, L. V., "Mechanical effects on endothelial cell morphology: in vitro assessment.," *In Vitro Cellular and Developmental Biology*, vol. 22, pp. 500–507, Sept. 1986.

[30] KASSIS, T., KOHAN, A. B., WEILER, M. J., NIPPER, M. E., CORNELIUS, R., TSO, P., and DIXON, J. B., "Dual-channel in-situ optical imaging system for quantifying lipid uptake and lymphatic pump function.," *Journal of Biomedical Optics*, vol. 17, p. 086005, Aug. 2012.

[31] KAWAI, Y., YOKOYAMA, Y., KAIDOH, M., and OHHASHI, T., "Shear stress-induced ATP-mediated endothelial constitutive nitric oxide synthase expression in human lymphatic endothelial cells," *AJP: Cell Physiology*, vol. 298, pp. C647–55, Mar. 2010.

[32] KORNUTA, J. A., "Code and schematics for ex-vivo lymphatic perfusion system (elps)." `https://github.com/jkornuta/elisha-evps`, 2012.

[33] KORNUTA, J. A., NIPPER, M. E., and DIXON, J. B., "Low-cost microcontroller platform for studying lymphatic biomechanics in vitro," *Journal of Biomechanics*, vol. 46, pp. 183–186, Jan. 2013.

[34] KUO, L., CHILIAN, W. M., and DAVIS, M. J., "Interaction of pressure- and flow-induced responses in porcine coronary resistance vessels.," *American Journal of Physiology*, vol. 261, pp. H1706–15, Dec. 1991.

[35] KUSHNER, D., "The making of Arduino. IEEE Spectrum." `http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino/`, October 2011.

[36] KVIETYS, P. R. and GRANGER, D. N., "Role of intestinal lymphatics in interstitial volume regulation and transmucosal water transport," *Annals of the New York Academy of Sciences*, vol. 1207 Suppl 1, pp. E29–43, Oct. 2010.

[37] KWON, S. and SEVICK-MURACA, E., "Noninvasive quantitative imaging of lymph function in mice," *Lymphatic Research and Biology*, vol. 5, no. 4, pp. 219–231, 2007.

[38] LEVICK, J. R. and MICHEL, C. C., "Microvascular fluid exchange and the revised Starling principle," *Cardiovascular Research*, vol. 87, pp. 198–210, June 2010.

[39] LIPOWSKY, H., "Microvascular rheology and hemodynamics," *Microcirculation*, vol. 12, pp. 5–15, Feb. 2005.

[40] MCHALE, N. and RODDIE, I., "Effect of transmural pressure on pumping activity in isolated bovine lymphatic vessels," *Journal of Physiology*, vol. 261, no. 2, pp. 255–269, 1976.

[41] MCHALE, N. G. and MEHARG, M. K., "Co-ordination of pumping in isolated bovine lymphatic vessels," *The Journal of Physiology*, vol. 450, pp. 503–512, June 1992.

[42] MIHARA, M., HARA, H., HAYASHI, Y., NARUSHIMA, M., YAMAMOTO, T., TODOKORO, T., IIDA, T., SAWAMOTO, N., ARAKI, J., KIKUCHI, K., MURAI, N., OKITSU, T., KISU, I., and KOSHIMA, I., "Pathological steps of cancer-related lymphedema: Histological changes in the collecting lymphatic vessels after lymphadenectomy," *PLoS ONE*, vol. 7, p. e41126, July 2012.

[43] MORARI, M. and LEE, J. H., "Model predictive control: past, present and future," *Computers and Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[44] MUTHUCHAMY, M., GASHEV, A. A., BOSWELL, N., DAWSON, N., and ZAWIEJA, D. C., "Molecular and functional analyses of the contractile apparatus in lymphatic muscle," *FASEB J*, vol. 17, no. 8, pp. 920–2, 2003.

[45] NELSON, T. S., AKIN, R. E., WEILER, M. J., KASSIS, T., KORNUTA, J. A., and DIXON, J. B., "Minimally invasive method for determining the effective lymphatic pumping pressure in rats using near-infrared imaging," *AJP: Regulatory, Integrative and Comparative Physiology*, vol. 306, pp. R281–R290, Mar. 2014.

[46] NIPPER, M. and DIXON, J. B., "Engineering the lymphatic system," *Cardiovascular Engineering and Technology*, vol. 2, no. 4, pp. 296–308, 2011.

[47] OHHASHI, T., AZUMA, T., and SAKAGUCHI, M., "Active and passive mechanical characteristics of bovine mesenteric lymphatics," *American Journal of Physiology*, vol. 239, pp. H88–95, July 1980.

[48] PENG, X., RECCHIA, F. A., BYRNE, B. J., WITTSTEIN, I. S., ZIEGELSTEIN, R. C., and KASS, D. A., "In vitro system to study realistic pulsatile flow and stretch signaling in cultured vascular cells," *AJP: Cell Physiology*, vol. 279, pp. C797–805, Sept. 2000.

[49] QUICK, C. M., VENUGOPAL, A. M., GASHEV, A. A., ZAWIEJA, D. C., and STEWART, R. H., "Intrinsic pump-conduit behavior of lymphangions," *American Journal of Physiology*, vol. 292, pp. R1510–8, Apr. 2007.

[50] RACHEV, A., DOMINGUEZ, Z., and VITO, R., "System and method for investigating arterial remodeling," *Journal of Biomechanical Engineering*, vol. 131, no. 10, p. 104501, 2009.

[51] RAHBAR, E., AKL, T., COTÉ, G. L., MOORE, JR, J. E., and ZAWIEJA, D. C., "Lymph transport in rat mesenteric lymphatics experiencing edemagenic stress," *Microcirculation*, Jan. 2014.

[52] RAHBAR, E. and MOORE, J. E., "A model of a radially expanding and contracting lymphangion," *Journal of Biomechanics*, vol. 44, pp. 1001–1007, Apr. 2011.

[53] RICHALET, J., RAULT, A., TESTUD, J. L., and PAPON, J., "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[54] ROCKSON, S. G., "Lymphedema," *The American Journal of Medicine*, vol. 110, pp. 288–295, Mar. 2001.

[55] ROCKSON, S. G. and RIVERA, K. K., "Estimating the population burden of lymphedema," *Annals of the New York Academy of Sciences*, vol. 1131, pp. 147–154, 2008.

[56] RUTKOWSKI, J. M., MARKHUS, C. E., GYENGE, C. C., ALITALO, K., WIIG, H., and SWARTZ, M. A., "Dermal collagen and lipid deposition correlate with tissue swelling and hydraulic conductivity in murine primary lymphedema," *The American Journal of Pathology*, vol. 176, pp. 1122–1129, Jan. 2010.

[57] SABINE, A., AGALAROV, Y., MABY-EL HAJJAMI, H., JAQUET, M., HÄGERLING, R., POLLMANN, C., BEBBER, D., PFENNIGER, A., MIURA, N., DORMOND, O., CALMES, J.-M., ADAMS, R. H., MAKINEN, T., KIEFER, F., KWAK, B. R., and PETROVA, T. V., "Mechanotransduction, PROX1, and FOXC2 cooperate to control connexin37 and calcineurin during lymphatic-valve formation," *Developmental Cell*, vol. 22, pp. 430–445, Feb. 2012.

[58] SAKAI, H., IKOMI, F., and OHHASHI, T., "Effects of endothelin on spontaneous contractions in lymph vessels," *American Journal of Physiology*, vol. 277, pp. H459–66, Aug. 1999.

[59] SCALLAN, J. P., WOLPERS, J. H., MUTHUCHAMY, M., ZAWIEJA, D. C., GASHEV, A. A., and DAVIS, M. J., "Independent and interactive effects of preload and afterload on the pump function of the isolated lymphangion," *AJP: Heart and Circulatory Physiology*, vol. 303, pp. H809–H824, Oct. 2012.

[60] SUN, D., HUANG, A., KOLLER, A., and KALEY, G., "Flow-dependent dilation and myogenic constriction interact to establish the resistance of skeletal muscle arterioles.," *Microcirculation*, vol. 2, pp. 289–295, Sept. 1995.

[61] SUN, Y., SUPPAPPOLA, S., and WRUBLEWSKI, T. A., "Microcontroller-based real-time QRS detection.," *Biomedical Instrumentation and Technology*, vol. 26, pp. 477–484, Oct. 1992.

[62] SWARTZ, M. A., "The physiology of the lymphatic system," *Advanced Drug Delivery Reviews*, vol. 50, pp. 3–20, Aug. 2001.

[63] TZIMA, E., IRANI-TEHRANI, M., KIOSSES, W. B., DEJANA, E., SCHULTZ, D. A., ENGELHARDT, B., CAO, G., DELISSER, H., and SCHWARTZ, M. A., "A mechanosensory complex that mediates the endothelial cell response to fluid shear stress," *Nature*, vol. 437, pp. 426–431, Sept. 2005.

[64] VENUGOPAL, A. M., STEWART, R. H., LAINE, G. A., DONGAONKAR, R. M., and QUICK, C. M., "Lymphangion coordination minimally affects mean flow in lymphatic vessels," *AJP: Heart and Circulatory Physiology*, vol. 293, pp. H1183–H1189, Mar. 2007.

[65] WEILER, M. J. and DIXON, J. B., "Differential transport function of lymphatic vessels in the rat tail model and the long-term effects of Indocyanine Green as assessed with near-infrared imaging," *Frontiers in Physiology*, vol. 4, pp. 1–10, Aug. 2013.

[66] WEILER, M. J., KASSIS, T., and DIXON, J. B., "Sensitivity analysis of near-infrared functional lymphatic imaging," *Journal of Biomedical Optics*, vol. 17, no. 6, p. 066019, 2012.

[67] ZAWIEJA, D. C., "Contractile physiology of lymphatics," *Lymphatic Research and Biology*, vol. 7, pp. 87–96, Jan. 2009.

[68] ZAWIEJA, D. C., DAVIS, K. L., SCHUSTER, R., HINDS, W. M., and GRANGER, H. J., "Distribution, propagation, and coordination of contractile activity in lymphatics," *American Journal of Physiology*, vol. 264, pp. H1283–91, Apr. 1993.

VITA

Jeff Kornuta is a Baton Rouge, LA native who grew up on a steady diet of football and crawfish. After graduating from Parkview Baptist High School in 2004, he pursued his Bachelor of Science in Mechanical Engineering at Louisiana State University, where he fell in love with both engineering and the Tigers. Graduating with his B.S. from LSU in 2008 with College Honors, Jeff continued his academic career working toward a Masters of Science in Mechanical Engineering at LSU. There he studied how momentum exchange tethers of various geometries affected heat loads during orbital reentry for human space missions, ultimately graduating with his M.S. in Spring 2009. In Fall 2009, Jeff was the first graduate student to join the Laboratory of Lymphatic Biology and Bioengineering (LLBB) under the direction of J. Brandon Dixon. In this capacity he was awarded a National Science Foundation Graduate Research Fellowship as well as a Georgia Institute of Technology President's Fellowship. Jeff plans on returning to the gulf coast to join Exponent, an international engineering and scientific consulting firm, in Summer 2014 as an Associate in Houston, TX.

Characterization of Lymphatic Pump Function in Response to Mechanical Loading

Jeffrey A. Kornuta

222 Pages

Directed by J. Brandon Dixon, Ph.D.

The lymphatic system is crucial for normal physiologic function, performing such basic functions as maintaining tissue fluid balance, trafficking immune cells, draining interstitial proteins, as well as transporting fat from the intestine to the blood. To perform these functions properly, downstream vessels (known as collecting lymphatics) actively pump like the heart to dynamically propel lymph from the interstitial spaces of the body to the blood vasculature. However, despite the fact that lymphatics are so important, there exists very little knowledge regarding the details of this active pumping. Specifically, it is known that external mechanical loading such as fluid shear stress and circumferential stress due to transmural pressure affect pumping response; however, anything other than simple, static relationships remain unknown. Because mechanical environment has been implicated in lymphatic diseases such as lymphedema, understanding these dynamic relationships between lymphatic pumping and mechanical loading during normal function are crucial to grasp before these pathologies can be unraveled. For this reason, this thesis describes several tools developed to study lymphatic function in response to the unique mechanical loads these vessels experience both *in vitro* and *ex vivo*. Moreover, this work investigates how shear stress sensitivity is affected by transmural pressure and how the presence of dynamic shear independently affects lymphatic contractile function.